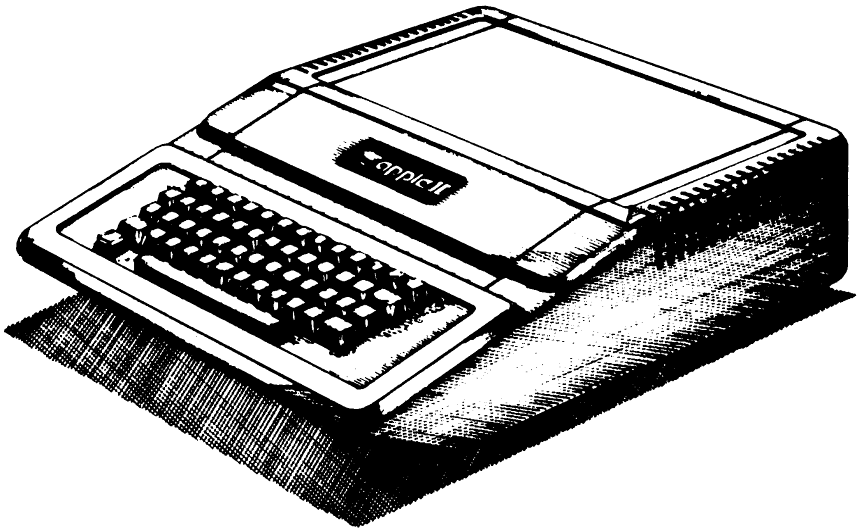
Apple 2 Computer Technical Information



Apple II Computer Documentation Resources

MAIN FOLDER

www.textfiles.com/apple/
18 September 2000

CATALOG

```
Name                 Type Crtr  Size    Flags       Last-Mod-Date      Creation-Date
-------------------- ---- ---- ------ ---------- ------------------ ------------------
'! T E X T F I L E S… TEXT MOSS  194K lvbspoImad    9/19/00  4:12 PM    9/18/00  7:09 PM
acos.hst.mod         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
advdem.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
aecomman.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
aids                 TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
alien.clues          TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
ansi.spcs            TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
apple.app            TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
apple.txt            TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
apple2.gs            TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
appleii.jok          TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
applemaf.txt         TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
applenet.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
apples.txt           TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
appleser.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
applesoft.tips       TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
appswitc.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
bin.ii               TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
bitsbaud.doc         TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
bootl-6              TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
bootl-6.hac          TEXT R*ch    97K LvbspoImad  10/29/99  8:20 AM    1/30/74  5:51 PM
catfur.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
catstuff.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cheat.app            TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cheats               TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
cheats.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cheats2.app          TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
copyprog.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
copyprot.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
correct.app          TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cr.adder             TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
crack1.txt           TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
crackdos.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
crackin.app          TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
crakowit.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cramit.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
cramit.txt           TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
crammin.app          TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
crisis.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
deathcheat           TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
diskgo.txt           TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
diskjock.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
dos.chart            TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
dosless.txt          TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
emu.pt.update        TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
errors.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
errors.txt           TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
expandca.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
futrae.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
icon.convert         TEXT R*ch    97K LvbspoImad    8/1/99 11:11 AM    1/30/74  5:51 PM
iigsprob.hum         TEXT R*ch    97K LvbspoImad    8/1/99 11:10 AM    1/30/74  5:51 PM
index.html           TEXT R*ch    97K LvbspoImad    7/13/00  8:11 AM    1/30/74  5:51 PM
joystick.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
kickmacr.app         TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
krack1.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
krack2.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
krack3.app           TEXT R*ch    97K LvbspoImad    8/1/99 11:09 AM    1/30/74  5:51 PM
```

```
krack4.app          TEXT R*ch    97K LvbspoImad     8/1/99 11:09 AM    1/30/74  5:51 PM
krack5.app          TEXT R*ch    97K LvbspoImad     8/1/99 11:09 AM    1/30/74  5:51 PM
krakowic.txt        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
krckwczt.app        TEXT R*ch   194K LvbspoImad     8/1/99 11:09 AM    1/30/74  5:51 PM
mac2info.app        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
maccrack.app        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
machine.app         TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
machinel.app        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
macteam.app         TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
memory.txt          TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
miffins2.txt        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
ml.part.i           TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ml.part.ii          TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ml.part.iii         TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ml.part.iv          TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ml.part.v           TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ml.part.vi          TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
oneguy.txt          TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
oo.world.info       TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
opcodez.app         TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
param2.app          TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
peekpoke.app        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
peeks.pokes         TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
peeks.pokes.1       TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
peeks.pokes.2       TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
peeks.pokes.3.1     TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
peeks.pokes.3.2     TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
pitfall2.txt        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
pm2600.app          TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
pokelist.app        TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
quick.draw.3        TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
quick.spells        TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
secretk.app         TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
softkey             TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
trace2.app          TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
usr.16.8k           TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
vidomac.app         TEXT R*ch    97K LvbspoImad     8/1/99 11:10 AM    1/30/74  5:51 PM
vt100               TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
wings.fury.cht      TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
wizardry.4.info     TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
xmodem              TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
ymodem.s            TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
zmodem.gbbs         TEXT R*ch    97K LvbspoImad     8/1/99 11:11 AM    1/30/74  5:51 PM
```

==
DOCUMENT ! T E X T F I L E S
==

Apple II Textfiles

www.textfiles.com/apple/
18 September 2000

With the introduction of the Apple II family of computers, the wonders of programming, communicating, and just plain geeking out became affordable for an entire generation of budding enthusiasts and their families. By the end of the 70's an entire culture had risen up around the Apple II, and the energy of thousands of hardware and software hackers went into learning every last op-code and settable switch within the machine.

It can't be discounted that Apple's successful foray into the educational market resulted in schools countrywide brimming with Apple IIs, and social groups collecting around the labs after school hours. All manner of things happened there, some documented below.

These files range from explicit memory maps of the Apple II to long tutorials on how to "crack" games, that is, remove all copy protection and make the game easier to distribute between other pirates.

Filename	Size	Description of the Textfile
DOCUMENTATION	DIRECTORY	"Soft Dox" for Apple Programs
GENIELAMP	DIRECTORY	Archive of the Genielamp A2, the GEnie Apple II Roundtable
WALKTHROUGHS	DIRECTORY	Walkthroughs of Apple II Specific Adventures
acos.hst.mod	6235	How to get Speed out of your HST and HST Dual Standard Modem on an Apple IIGS
advdem.app	16645	Technical notes for Advanced DeMuffin II, a cracking tool
aecomman.app	1792	A list of commands for Ascii Express
aids	1024	Method for detecting the "Cyberaids Virus", by The Chemist
alien.clues	1448	Passwords for Alien Mind, by The Undertaker and the Vandal
ansi.spcs	24911	ANSI and VT100 Codes
apple.app	4157	Combining Applesoft with Assembly Language
apple.txt	4189	The Text of the Apple-Microsoft Agreement
apple2.gs	9388	The Sad, True Truth of the Apple II GS (Stands for Goddamned Slow)
appleii.jok	1384	The Unofficial Apple II Brainwash Test by Fred E. Long
applemaf.txt	22452	The Apple Mafia Story, as Told to Red Ghost
applenet.app	4096	Advertisement for Apple-net software. Note feature list
apples.txt	8230	Why the Apple II is Broken
appleser.app	11205	Apple //c Serial Port Information
applesoft.tips	2320	The Beagle Brothers Applesoft Tips Guide
appswitc.app	2677	Apple //e Soft Switch, Status, and other I/O locations
bin.ii	18944	Apple II Binary File Format, developed by Gary B. Little
bitsbaud.doc	11553	Bits, Baud Rate, and BPS, by michael A. Banks,

		1988
bootl-6	102420	Collection of Apple-0riented Texts and Flotsam from the Early 1980's.
bootl-6.hac	102420	Bootlegger Magazine Excerpts (Apple II Stuff)
catfur.app	7176	Bit Blaster's Information on the Cat Fur Modem
catstuff.app	9818	Expanding your Apple Cat // by the Warewolf
cheat.app	4424	All manner of cheats for various Apple II games
cheats	7416	LARGE Collection of Apple Cheats (Break into Monitor and Modify)
cheats.app	2749	The Penguin's Apple Cheats
cheats2.app	4498	Apple Pirate's Cheats
copyprog.app	2991	How to Copy Programs, by the Three Musketeers
copyprot.app	15163	Copy-Protecting your own disks, by Thomas T. Brylinski
correct.app	5716	Corrections to programming for the Apple Cat
cr.adder	1441	How to add Carriage Returns to Appleworks Databases
crack1.txt	1023	Introduction to a Talk on Software Piracy
crackdos.app	15403	Introduction to how AppleDOS operates
crackin.app	9989	An introduction to cracking by The Necromancer
crakowit.app	3647	Kracowicz' Kracking Corner IV
cramit.app	5062	An Introduction to Program Compression
cramit.txt	7040	Some Tips on Cramming Data with an Apple
crammin.app	5071	A simple compression scheme
crisis.app	1900	How to crack Crisis Mountain, by Doctor Who
deathcheat	517	Cheat for "Death Sword"
diskgo.txt	613	Getting Faster Apple DOS Speeds by Tamerlane of the Ring
diskjock.app	51504	Examining protected Applesoft programs, by the Disk Jockey
dos.chart	1678	The DOS 3.3 Memory Access Chart
dosless.txt	1792	Creating an Apple DOS-Less Disk
emu.pt.update	3739	Message: Bugs in IIGS Proterm v1.9p
errors.app	4286	A comment on error traps, by Nick Fotheringham
errors.txt	4480	A Comment on Error Traps by Nick Fotheringham from the Apple Barrel
expandca.app	9367	Expanding your Apple Cat, by Warewolf
futrae.app	4684	The Future Evolution of Ascii Express (Humor)
icon.convert	3308	Converting Apple IIGS Icons to Clip Art by Marty Knight
iigsprob.hum	2680	The Apple IIgs Sound Problem
joystick.app	5961	The Official Joystick Review Guide, by The Tracker
kickmacr.app	9981	How to kick butt with AE Macro Action
krack1.app	2927	High Technology's Cracking Tutorial, Part I
krack2.app	1765	High Technology's Cracking Tutorial, Part II
krack3.app	2239	High Technology's Cracking Tutorial, Part III
krack4.app	1887	High Technology's Cracking Tutorial, Part IV
krack5.app	2560	High Technology's Cracking Tutorial, Part V
krakowic.txt	13198	Kracowicz' Cracking Tips from ROM Radier
krckwczt.app	137510	The Kracowicz Basics of Cracking Series. A++
mac2info.app	11449	Late-breaking (1987) information on The Macintosh II
maccrack.app	5981	The Byte's introduction to Mac Cracking
machine.app	13084	Black Bag's Introduction to Machine Language for Cracking
machinel.app	15408	Dr. Firmware's Tutorial of Machine Language
macteam.app	9569	Macteam's thoughts on copy protection on the

		Macintosh
memory.txt	12020	An Apple Peek Poke, Call List
miffins2.txt	1421	How to use Demuffin Plus
ml.part.i	5680	The Machine Language Tutorial Disk by Dr. Firmware
ml.part.ii	5370	The Machine Language Tutorial Disk Part II by Dr. Firmware
ml.part.iii	5627	The Machine Language Tutorial Disk Part III by Dr. Firmware
ml.part.iv	4970	The Machine Language Tutorial Disk Part IV by Dr. Firmware
ml.part.v	5703	The Machine Language Tutorial Disk Part V by Dr. Firmware
ml.part.vi	5210	The Machine Language Tutorial Disk Part VI by Dr. Firmware
oneguy.txt	1408	Hey, If You Pirate the Game, Don't Call Tech Support
oo.world.info	3206	The Magnet Previews Out of This World GS
opcodez.app	2811	Various Apple Opcodes
param2.app	16201	Parameters of Nibbles Away II for various software packages
peekpoke.app	21120	A really large collection of Apple II PEEKs and POKEs
peeks.pokes	2957	Description of the differences between CALL, PEEK and POKE in Applesoft
peeks.pokes.1	6166	Collection of Apple Peeks and Pokes
peeks.pokes.2	4396	Collection of Apple Peeks and Pokes in the Zero Page Area
peeks.pokes.3.1	14869	Apple Peeks, Pokes and Calls List Version 2.1 by The Enforcer (May 1984)
peeks.pokes.3.2	5377	Miscellaneous Applesoft Information, by Control Reset
pitfall2.txt	2176	Soft Docs for Pitfall 2: Lost Caverns
pm2600.app	3045	The Poor Man's 2600 Hertz by Sir Briggs
pokelist.app	19769	A really large collection of Apple II PEEKs and POKEs (Duplicate)
quick.draw.3	5122	Quick-Draw Adventure Mapper by Sherlock Apple (Part III)
quick.spells	3256	Quick-Draw Adventure Mapper by Sherlock Apple (Spells)
secretk.app	6956	Secret Keys: Little easter eggs and news about Apple II games
softkey	21083	Softkey Unprotections for a Variety of Commercial Programs
trace2.app	11562	Mr. Xerox' boot tracing, volume I (badly converted)
usr.16.8k	85773	The Info File on the USR Robotics 16.8k Model
vidomac.app	33057	1986 Seminar on "Macintosh in Film and TV Production"
vt100	3685	DEC VT-100 Compatible Cursor Command Sequences
wings.fury.cht	606	Cheat to Wings of Fure
wizardry.4.info	3012	Advice about playing Wizardry IV
xmodem	21581	XMODEM Protocol Reference, by Ward Christensen January 1, 1982
ymodem.s	13048	YMODEM Source Code for GBBS by Mike Golazewski or Greg Schaefer
zmodem.gbbs	7045	The Addition of ZMODEM to GBBS!

There are 98 files for a total of 1,155,472 bytes.
There are 3 directories.

If you wish to have the entire directory conveniently archived and compressed into one file, please download either apple.tar.gz (6130920 bytes) or apple.zip (6496886 bytes) instead of all the files separately.

###

```
================================================================================
DOCUMENT acos.hst.mod
================================================================================
```

How to get

S P E E D

out of your HST and HST Dual Standard Modem

using

ACOS Version 2.01d5

and an Apple IIGS

```
-------------------------------------------------------------------------------
    Brought to you by The Oggman, creator of OGG-Net Networking Systems
          Call Infinity's Edge (415) 820-9401 or any OGG-Net BBS
-------------------------------------------------------------------------------
```

HST Basics

Those of you who have spent big bucks on the HST modem, hoping to get 14.4K bps, have probably been disappointed, misled, or both. When used with ACOS, the HST will only get throughput of 9600 bps with NO COMPRESSION. Why is this? Well, its actually pretty simple.

When you're dealing with high speed modems, you have to differenciate between "connect rate" and "DTE rate." Connect rate is the speed in which both modems are talking with each other. You can get the connect rate by looking at the number after the CONNECT message (ig 1200, 2400, 9600) or by looking up the numeric result code. DTE rate is the rate at which the serial port is set at when it makes the call.

HST's can be communicated to with DTE rates of up to 38400 bps. This means that, even though its only (!) a 9600 bps modem (so to speak), you can actually send commands to it at baud rates of up to 38400. In normal operations, the DTE rate will drop down to the connect rate as soon as the modem completes a call to the other modem.

Compression and 14.4K

In order to use V.42 or MNP Level 5 data compression, THE DTE RATE HAS TO BE HIGHER THAN THE CONNECT RATE. Likewise, to get actual throughput of higher than 9600 bps (12K, 14.4K), the DTE rate will have to be higher than 9600. So, for the Apple, this generally means your serial port will HAVE TO STAY AT 19,200 BAUD.

Do you see the problem? ACOS autobauds the serial port to whatever the connect rate is. If someone connects at 9600 baud, then the serial port is set to 9600 baud, making data compression and high speed totally useless. So, what to do about this problem?

Making the Change

To make your HST operate at its optimum efficiency, you'll have to fix the DTE rate of your modem and make a little change to your ACOS.OBJ file. First, run the CONFIG.SYS program and choose GS Modem Port and the HST modem (sorry, I haven't deciferd what to do for the SSC yet. Maybe later). Change the init string so &D2 is replaced by &B1 (&D2 does absolutely nothing), and the X6 is changed to X4. &B1 will fix your modem's DTE rate. Pick 19200 for the default baud rate.   You should also make sure your NVRAM settings include &K1 and &H1. These settings are needed for data compression and flow control.

Now, load BASIC.SYSTEM and type the following:

```
bload acos.obj,a$800            (loads the ACOS.OBJ file)
call -151                       (enter monitor)
ff0:ea ea ea ea ea ea           (make the change)
bsave acos.obj,a$800,l$5300     (save it)
```

Now, all you have to do is run your board and you're set. The modem will fix its DTE (the &B1) at 19200 (the default baud rate). When someone calls, ACOS will no longer reset the DTE rate to the connect rate, but keep it at 19200. Info(2) will still contain the connect rate/300, however.

Warp Speed on the HST

So, now we have our HST's zooming along with a fixed DTE rate of 19200. This is the best we can get, right? Well, we know that the HST will operate at DTE rates of up to 38400 baud, but the Apple IIGS serial port will only go to 19200. Or will it?

Well, very recently, Apple guru Guy T. Rice came out with a little CDA called Fastport GS 38.4 that will actually open up the Apple IIGS modem port at 38400 baud. It appears that 19200 is only a firmware limit, not a limit of the SCC chip. After decifering Mr. Rice's little desk accessory, I mananged to incorporate it into an ACOS mod that will actually fix the DTE rate of the modem port to 38400, allowing the maximum throughput of the HST for ACOS 2.0d5 bulletin boards.

WARNING: Even with a 7 mhz Transwarp GS card installed, 38400 baud is VERY fast and you'll probably get dropped characters. This might not be a problem once we get 10 or 13 mhz out of our amazing machines, but until then, do this mod with caution. I personally have found no problem with this mod on my 6.25 mhz GS.

First, run CONFIG.SYS just as above making the same changes. Run BASIC.SYSTEM and type the following:

```
bload acos.obj,a$800                            (loads the ACOS.OBJ file)
call -151                                       (enter monitor)
ff0:ea ea ea ea ea ea                           (make the change)
ffd:20 30 10 ea ea ea ea ea ea ea ea ea ea
1030:78 a9 0c 8d 38 c0 a9 01 8d 38 c0 a9 0d 8d 38 c0 a9 00 8d 38 c0 60
bsave acos.obj,a$800,l$5300                     (save it)
```

Calling Out

As a BBS user, you will also have to do your part in getting the best throughput for your HST.

First, you will also have to fix your modem's DTE rate to the highest baud rate possible (19200 normally, 38400 using the Fastport CDA). To do this, set your baud rate at its highest level, and type AT&B1&K1&H1 and return while in the terminal mode of your term program.

Now for the hard part. Proterm and most every other term program for the Apple has the same problem that ACOS has; it autobauds. I don't have a quick fix for this, but you can get around it. Just call the HST board with a baud rate of 19200 (or 38400). It will connect at 9600 like normal. Once it connects, however, change the baud rate back to what it was. In Proterm, do an OA-O and put it back to 19200. If you have the Fastport CDA installed, activate it once you connect with the HST board.

That should do it. If you have any questions, I can be reached at my board Infinity's Edge (user #1) or through any board in the OGG-Net network.

Paul Parkhurst
The Oggman

```
=============================================================================
DOCUMENT advdem.app
=============================================================================

                   TECH-NOTES FOR
                 Advanced Demuffin 1.1

                    Written  by
                     The Stack

                   Copyright 1983
                   Corrupt Computing

If you want source code for any of Corrupt Computing's line of utilities, just
contact THE INSPECTOR on THE TWILIGHT PHONE.
===============================================================

ZERO PAGE LOCATIONS

  $22  WNDTOP     These 2 zero page locations, WNDTOP and WNDBTM,
  $23  WNDBTM      are used so that the character that the
             character output routines in the monitor will
             output characters only in the window below the
             first 3 lines and above the bottom 2 lines.
             The top 3 and bottom 2 lines are used for title
             lines and status display.  These locations
             should be restored to normal upon return from
             your RWTS if it uses them, although most RWTS's
             don't use these reserved monitor locations.

  $26  GBASL      These 2 zero page locations are used by many
  $27  GBASH       routines throughout Advanced Demuffin, such as
             the PRINT routine and the routines to display
             the status codes on the disk map, but they do
             not need to be saved before going to your RWTS.
             Many RWTS's, including RWTS 3.3, use these
             locations in several places.

  $36  CSWL CSWL and CSWH should always point to the address
  $37  CSWH  of the current character output routine.
             Advanced Demuffin sets these locations to point
             to $FDF0, the standard character output
             routine.  Note that the outputed characters
             will no longer go through DOS as there may be
             no DOS in the machine.  Advanced Demuffin
             changes the contents of these locations to
             point to $Cx00 when a number from 1-7 is
             pressed during a conversion or after a
             conversion is completed, where x is the number
             pressed.  These locations should be restored to
             point to $FDF0 if your RWTS uses them in any
             way.  Most RWTS's, including RWTS 3.3, don't
             use them at all.

  $4A  TEMP1      Although most RWTS's don$, use these locations,
  $4B  TEMP2       they are used as scratch locations by Advanced
  $4C  TEMP3       Demuffin and are VERY IMPORTANT!  Be sure and
```

```
             save them if your RWTS even looks at them.  The
             most important location to save is $4B, which
             contains the page number that the current
             sector is being loaded into.  Note that this is
             a duplicate of the X register upon entry into
             the user's IOB module at $1400.

PRE-PROGRAM NON-ZERO PAGE LOCATIONS

 $200  BUF  Page 2, the character input buffer, is used as a
             buffer to hold the file name of the RWTS or IOB
             module to be loaded.  This page may be used by
             your RWTS, but your RWTS may not reside in the
             area between $200-$21E (unless you don't plan
             on loading anything), as this portion of page 2
             will be destroyed upon a load.

 $3F2  RESET      Advanced Demuffin sets this pointer to pnint to
             $FF59.  This means that whenever the RESET key
             is pressed, the Apple will jump into the
             monitor.  If this is not desired, $12C9 (low
             byte) and $12CE (high byte) may be changed to
             have the RESET key go wherever you want it to
             go including $801 (Advanced Demuffin entry).
             $12C9 normally contains a $59 and $12CE
             normally contains an $FF.

 $3F5  AMPVEC     Advanced Demuffin sets up these locations to
 $3F8  CTYVEC      jump to the Advanced Demuffin entry point
             ($801) when Applesoft recieved the "&" command
             and when the monitor recieved the CTRL-Y
             command.  This provides a useful way to get
             back into Advanced Demuffin after exit.

 $400-$7FF  Many times Advanced Demuffin displays data and
            status marks on the screen by storing this data directly into
            this area of memory.  This includes all marks on both the track
            map and the disk map as well as numbers on the bottom screen
            line, and dashes and other messages on the 3rd and 23rd lines.

INTER-PROGRAM LOCATIONS

 $800       This is the location where Advanced Demuffin is
             designed to run at.  This location contains an
             $EA (NOP) as the byte at $800 is often replaced
             by a $00.  This is NOT the entry point to
             Advanced Demuffin ($801 is the entry) although
             if there is an $EA here it won't make any
             difference if you use this as the entry.

 $801  START0     This is the entry point to Advanced Demuffin 1.1
             where there are two instructions, SEI and CLD,
             before the actual START of Advanced Demuffin.

 $803  START      This is the actual start of the program which
             sets CSWL and CSWH to point to the monitor
             routine COUT1, sets the RESET, AMPVEC, and
             CTYVEC as mentioned above (see appropriate
             label), sets the full screen as a window except
             for the top 3 and the bottom 2 lines, clears
             the screen, puts the title at the top, the
             status line at bottom, and starts off the
             program by displaying the menu.

 $F1E  IOB  This is the IOB that Advanced Demuffin uses when
             it uses RWTS.    The built-in IOB module (IOB33)
             which is described below, as well as the
             default user IOB module (at $1400) also use
             this IOB.  The default contents of this IOB are
             described in detail below:

 $F1E:01 60 IOB    DFB $01,$60
 $F20:01    DRIVE  DFB $01
 $F21:00    VOLUME       DFB $00
 $F22:00    TRACK  DFB $00
 $F23:00    SECTOR       DFB $00
 $F24:2F 0F        DW DCT
 $F26:00    DPAGL  DFB $00
 $F27:80    DPAG   DFB $80
 $F28:00 00        DFB $00,$00
 $F2A:01    CODE   DFB $01
 $F2B:00    ERROR  DFB $00
 $F2C:00 60 01           DFB $00,$60,$01
 $F2F:00 01 DCT    DFB $00,$01
 $F31:EF D8        DFB $EF,D8

Note that the slot number used by Advanced Demuffin could easily be changed bu
changing $F1F to the $x0 where x is the slot number of the desired drive.

 $F33  IOB33      This is the built-in IOB module used to write to
             3.3 formatted disks.  A disassembled listing of
             it is included below:

 $F33- IOB33  STY SECTOR    ;Store sector
 $F36-            STX DPAG        ;and page number
 $F39-            LSR A     ;Convert phase # to track #
 $F3A-            STA TRACK     ;and store it
 $F3D-            LDA DRV         ;Check # of drives
 $F40-            STA DRIVE     ;and store it as drive to write to
 $F43- THERE  LDA #2        ;Set command code to write
 $F45-            STA CODE        ;and store it
 $F48-            JSR GORWTS    ;and go to 3.3 RWTS to write it
 $F4B-            LDA #1          ;Restore read
 $F4D-            STA CODE        ;command code
 $F50-            LDA ERROR     ;Check for an error
 $F53-            BCC RTS4        ;Exit if none
 $F55-            CMP #$10        ;Write protect error?
 $F57-            SEC       ;Keep carry set
 $F58-            BNE RTS4        ;Not write protect, exit w/carry set
 $F5A-            LDY #$27        ;Display write protected
 $F5C- MOV4   LDA WPER1,Y   ;error message
 $F5F-            STA SCLN1,Y   ;an``xsk whether
 $F62-            LDA WPER2,Y   ;to continue or
 $F65-            STA SCLN2,Y   ;start over
 $F68-            DEY
 $F69-            BPL MOV4
 $F6B-            JSR PRINT     ;Print 3 beeps
 $F6E-            DFB $07,$07,$87
 $F71- KEY10  JSR KEYIN     ;Read a key - go back to menu if esc
 $F74-            CMP #$C3        ;Continue?
 $F76-            BEQ CONTIN    ;Yes, branch
 $F78-            CMP #$D3        ;Start over?
 $F7A-            BNE KEY10     ;No
 $F7C-            PLA       ;Yes
 $F7D-            PLA)      ;Pull return address off stack
 $F7E-            JSR REPLNS    ;Replace top 2 lines w/ title lines
 $F81-            JMP GOTVAL  `Ao   ?xtts over
 $F84- CONTIN JSR REPLNS
 $F87-            BMI THERE     ;Always taken
```

```
$13FA-$13FB These 2 bytes are unused

$13FC-$13FF These 4 bytes are reserved for the address and
             the length of the IOB module 8^sn it is being
             loaded.  Advanced Demuffin loads the first
             sector from the track/sector list of the IOB
             module at $13FC.  Since the first 4 bytes of
             this sector contain the address and the length
             of the file, those bytes reside in these
             locations.  Therefore, the actual IOB module
             will start at $1400 (just below).
$1400  IOBM This is the user IOB module.  The LOAD NEW IOB
             MODULE will load a file into thi
s area (see
             above).  A disassembled listing of the default
             user IOB module is included in the main manual.

$1419-$14FB These bytes between the user IOB module and RWTS
             3.3 are left free for an IOB module longer than
             the default one.  This allows an IOB module to
             take up as much as $FC bytes total.

$14FC-$14FF These 4 bytes are unused.

$1500-$1CDB RWTS 3.3 resides in this area of memory.  It is
             just standard RWTS that has been relocated to
             run at this address.  Advanced Demuffin uses
             the entry at $1A00.
```

Below are some other locations used as scratch by Advanced Demuffin. These may be looked at by your IOB module in determining various options about how it is to read sectors from the source disk if desired.

```
$1CE0  SCVER      This location contains either a $0C or a $0F
             for 13 and 16 sector modes, respectively.

$1CE1  STPHS      This location contains phase number to start
             reading data from the disk with.  It defaults
             to $00.  (Since it is a phase #, a $01 would
             mean track .5, etc.)

$1CE2  ENPHS      ENPHS is the same as STPHS except that it
             contains the last phase to read data from.
```

$1CE3 STSEC   STSEC contains the first sector within the phase specified by STPHS that data should be read from.

$1CE4 ENSEC   ENSEC contains the last sector within the phase specified by ENPHS that data should be read from.

$1CE5 CRPHS   This location contains the current phase that data is being read from.

$1CE6 CRSEC   This location contains the current sector that data is being read from.

$1CE7 BGSEC   BGSEC contains the sector number within the phase specified by BGPHS (below) that data has started being read from this pass. i.e. If you are converting an entire 16 sector disk with the default options and the default buffer size ($70 pages), during the first pass BGPHS and BGSEC will both contain a $00 (phase 00, sector 00 was the start phase, sector in this pass). During the second pass, BGPHS and BGSEC will contain $0E and $00, respectively. (The second pass started with track 07, sector 00 and track 07 is phase $0E).

$1CE8 BGPHS   See above.

$1CE9 BYPHS   This byte contains the increment in phases. i.e. The default increment, 1.0, would be $02.

$1CEA NRETRY   This byte contains the maximum number of retries (normally $01).

$1CEB RETRY   This byte is used as a counter counting down from the maximum number of retries to $00. On the first attempt to read a sector, RETRY will equal NRETRY.   If the carry is set upon return from the user's IOB module, RETRY will be decreased. If it is less than zero, a read error will result. If not, a read will be re-attempted. This process will continue until the sector either reads correctly or until RETRY is less than zero.

$1CEC DRV This location contains either a one or a two respective to the number of drives being used. The built-in IOB module uses this location to determine which drive to write data to.

$1CED-$1CEF These 3 bytes are unused

$1CF0 BUFST   BUFST contains the page number of the start of the buffer. This buffer is used to store data read off the source disk. By changing this location and/or BUFEN (below) you can easily change the buffer size and the location of

```
             Advanced Demuffin's buffer.  This location
             normally contains a $20 meaning that the buffer
             normally starts at $2000.

$1CF1  BUFEN      BUFEN contains the page number of the first page
             not to be included in Advanced Demuffin's
             buffer (see above).  i.e. If this location
             contained a $90 (the default value) and BUFST
             (see above) contained a $20 (the default again)
             the buffer would reside from $2000 to $8FFF
             (which it normally does).  However, this
             byte may be changed from a $90 to another
             value, such as a $B8, making the buffer much
             larger.  In this example, your buffer would be
             $9800 bytes long!  This will, of course, erase
             DOS when you attempt to convert the disk; but
             no problem - Advanced Demuffin does not require
             DOS anyway. (Not even for loading RWTS and IOB
             modules!)  Another use for changing this byte
             the one before it is to move the buffer to a
             different place.  i.e. If you had a hi-res
             screen on hi-res page 1 ($2000-$3FFF) that you
             wanted to keep in memory, you could simply
             change BUFST ($1CF0) to $40, forcing the buffer
             to start at $4000 instead of $2000 - saving
             your screen.

$1F00  DIRSEC     This page is used as a scratch page when loading
             sectors from the disk.  i.e. When loading a
             RWTS or an IOB module, the directory sector
             containing the name of the file to load will be
             read into this page.  The track and sector of
             the track/sector list will be found and the
             track/sector list will then be loaded here.

$BD00  USRRWTS    This is address JuMPed to by the default user
             RWTS.      You should either have an RWTS here or
             the IOB module should be changed to point to a
             different location.  Note the $BD00 does not
             necessarily have to be the start of the RWTS
             when using the default user IOB module, it must
             be the ENTRY POINT of the RWTS.  In fact, most
             RWTS's have a STARTING ADDRESS of $B800 but an
             ENTRY POINT at $BD00.  Keep this in mind when
             you load an RWTS module from disk.

$C000  KEYBD      These are the only hardware locations used by
$C010  KEYCLR      Advanced Demuffin other than during the screen
             dump where $Cx00 is JSRed to (where x is the
             slot number).

The following monitor routines are used by Advanced Demuffin:

$F847  GBASCALC
$FB2F  INIT
$FC58  HOME
$FD8E  CROUT
$FDED  COUT
```

```
$FDF0  COUT1
$FF59  MONITOR
```

ACTUALLY DEMUFFINING A DISK:

Using Castle Wolfenstein as an example. (I used this because it is the only thing that I that have that wasn't cracked!)

1) Boot up Castle Wolfenstein.   Before the cursor appears press CTRL-C. The one character buffer in the keyboard will remember it and when DOS asks for a character it will give the CTRL-C. The CTRL-C will cause Castle Wolfenstein's hello program to break into BASIC after it is loaded.

2) Enter the monitor with "CALL-151". Enter "4000<B800.BFFFM" this will move MUSE's RWTS down to a "safe" area of memory.

3) Insert a "slave" disk in drive one and boot the disk with 6 CTRL-P (If your disk drive is in slot 6 of course).   Press RESET when the prompt (]) appears. This will prevent your "hello" program from erasing MUSE's DOS.

4) Insert a disk with at least 10 free sectors on it. Save out the RWTS with "BSAVE MUSE-RWTS,A$4000,L$800".

5) Brun Advanced Demuffin. Move the light bar to "LOAD A NEW RWTS MODULE" and press RETURN.

6) Type the page number to load the RWTS at ($B8). Then type the file name that you saved it under and press RETURN.

7) Move the light bar to "CONVERT DISK" and press RETURN. You do want to change default values.

8) The disk is a thirteen sector disk, so enter a "3" for the question "SECTORS PER TRACK? (13/16)".

9) You want to copy from track $03,sector $00 to track $22, sector $0c. The increment is $1. (you are copying from track $03 because you don't need MUSE's DOS.)

10) You might encounter some errors, so use "1""as the number or retries.

11) If you have two drives in the same slot, enter a "2" for drive to copied to.

12) If you only have one drive, enter a "1" for drive to be copied to.

13) Insert the proper disk(s) when Advanced Demuffin prompts you.

Advanced Demuffin will then start converting the disk.   After the disk has been
converted, and Advanced Demuffin displays this message "PRESS ANY KEY TO CONTINUE", you should write down all the sectors that read errors on them. If you have a printer, all you have to do is press the slot number of the printer, and Advanced Demuffin will dump the screen to the printer.

You should then re-convert the sectors that had read errors (use at least 2-retries). If those sectors don't convert this time, they are probably just un-written DOS 3.2 sectors.

Use Super Copy III or Master Create to copy dos on to the target disk. You
should then rename the "hello" program from ^HELLO to HELLO.

You should then have (hopefully) a cracked copy of Castle Wolfenstein!

```
==============================================================================
DOCUMENT aecomman.app
==============================================================================

]I ]I^R]I^R]I]I[ I     ]I  (I         #.     ]I    ]I  I  10

---------------------------------------
[Ctrl-S pauses/Space=quit]

<=======================================>
<=========== THE OUTLAW ==============>
<=== BRINGS ==========================>
<=== YOU =============================>
<=======================================>
<========ALL THE COMMANDS=============>
<=======================================>
<===========FOR...AE=================>
<=======================================>


 CRTL Q...EXITS TERMINAL MODE AND IS
NEEDED TO BE ENTER BEFORE ANY OF THE
FOLLOWING COMMANDS:

+>
  ? HELP
  % LET'S YOU RUN INSTALL
  O LET'S YOU SAVE INFO TO DISK WHEN
    BUFFER IS FULL
  P TURN ON OR OFF PRINTER
  L LOAD FILE INTO BUFFER
  1 SHOWS MENU 1
  2 SHOWS MENU 2
  D LET'S U DIAL
  K THIS LET'S YOU CHAT
  H LET'S U HANG UP
  X EXIT PROGRAM
  S U/L A PROGRAM
  J VIEW A PROGRAM
  V VIEW BUFFER
  G D/L A PROGRAM
  C TO CLEAR BUFFER
  R COPY INCOMING DATA ON/OFF
  W WRITE BUFFER TO DISK
  I ALLOWS YOU TO CATALOG, OR DELETE
  ! DISPLAYS PROGRAM STATUS
  F FREE BUFFER SPACE
  M MACRO SELECTION
  - DISPLAY PREFIXED CHARACTERS
  Z SCREEN FORMAT WRAP/TRUNK
  B BAUD RATE
  A SHOW CONTROL CHARACTERS
  Y EDITOR
  U UPDATE MACRO/ALSO W/C BUFFER A TIME
  " KEYCLICK ON/OFF
  + AUTO ANSWER
  / DO CRC
  $ EMULATION MODE
  # BRIEF RUN MODE
  ' ANSWER BACK ON/OFF
  E DUPLEX H/F
  N SET DELAY
+>.....................................
......................................>
E/MAIL

TO

<==THE OUTLAW==>

FOR MORE HELP.



---------------------------------------

Enter (1-10, M=Menu, Q=Quit) :
```

```
==============================================================================
DOCUMENT aids
==============================================================================
```

This Small basic Program will Allow you to detect if the "CyberAids Virus" has infected any or some of your system files. This will only detect for the specific "CyberAids Virus" found on Z.link Plus and a now Increasing rapidly amount of other simuliar infected sys type files. Note: "CyberAids Virus" does NOT infect Prodos.You may further check your root volume by the following commands:

```
              ]bload /yourvol,a$2000,tdir
              ]?peek(8703)
              ]0  <-- This should be the result!!
```

If the result is any number other than 0 than its a good guess that your volume has been infected. You can save it by Block editing $21ff back to 0...Maybe.

I hope this helps. Any comments and suggestions are welcome.

The Chemist
The Lab bbs
604-Labhaha

Sending File
Sending BatchHeaderBlockDataFinalAckStartupNameEOFTimeoutMismatchFailureLast Send Was OKLast Send FailedL LbL∑,∑”∑
∑∑≠†∑≠∑∑±¿∑ ÃF ZWaiting for Receiver∑,∑∑n∑„∑∑o∑∑pdq §LÀ
∑ˆ±Æ∑Â¯∑˘°∑∑¥∑)Ì∑°∑∑

```
================================================================================
DOCUMENT alien.clues
================================================================================
```

..The Pirates Hold..

From The Undertaker and the Vandal;

A little help for those that need it!
The Passwords to all the terminals on the first 6 levels.
These aren't clues, but the real thing.
Clue list later, I am having too much fun to worry about the rest of you now!

Alien Mind Password List;

Clues;	Passwords;
1. Something I sent you.	Telegram
2. Color of ancient Earth's seas.	Aqua
3. A bed for a lazy afternoon.	Hammock
4. Something a sun gives off.	Radiation
5. What do you need info on?	Elevator
6. Name of Aaron's Wife.	Judy
7. Ancient Seductress.	Siren
8. Greeting.	Biologist HO!
9. A scale (musical).	CDEFGAB
10. Goes thru water without getting wet.	Light
11. Who to call.	Bio-Lab
12. A musical Hat.	Sombrero
13. A liquid that holds water.	Glass
14. Roman numeral for 1,174.	MCLXXIV
15. An Audio tool.	Ear
16. A visual enhancer.	Telescope
17. Legendary Continent.	Alantis
18. Project we last worked on together	Transit
19. Life giving fountain	

```
===============================================================================
DOCUMENT ansi.spcs
===============================================================================

_O

 Number       : 17 of 21                              [ ACOS/MACOS Programming ]
 Subject      : ANSI/VT-100 CODES
 Addressed to : ALL
 Author       : Nuke at The Space Bar  (#198@#4)
 Date Posted  : Thursday, January 13th, 1994  20:57:51 PST
 Size         : 24632 bytes
-O
Somebody was asking for the VT-100 codes, well here is a listing of boith
ANSI/VT-100 codes from the Supertac docs I have.....hope this helps you out??

------------------------------------------------------------------------------
SuperTac Tech note #2 - ANSI & DEC VT100 CODES...
                        Oct 30, 1990 - Larry Hawkins

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v
                         ANSI & DEC VT100 Codes
^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

   The following text will describe the various escape codes that allow
computers to transmit color text over communication lines.  Using ANSI
escape codes, any computer can receive and transmit color text as long as
the communication program can interpret the escape sequences.
   The escape sequences can be thought of the same as ones sent to a printer
to change the appearance of the output.  All ANSI codes begin with the one
byte character ESC (decimal 27), and are followed by the left bracket "[".
Additional parameters, which follow the bracket are seperated with a semi-
colon.All codes are ended with a single alphabetic character which determines
the function of the escape sequence.
   Since the characters come in one at a time, as soon as the ESC character
is received start building the sequence until an alphabetic character is
input.The case of the alphabetic character is very important since they
mean different things.   For example,"H" (which means set cursor position)
is different than "h" ,(which means set the display width and type).

   Notes:
       1) The default value is used when no explicit
          value is given, or a value of zero, is
          specified.

       2) The default value is 1 unless otherwise
          specified below.

       3) # - Numeric Parameter.  A decimal number
          specified with ASCII characters.

       4) In the control sequences described below,
          ESC is the 1 byte code for ESC (decimal 27),
          and not the three characters "ESC".

CURSOR CONTROL
```

```
Cursor Position (CUP)
  ESC[#;#H          Moves the cursor to the position
                specified by the parameter.  The first
                parameter specifies the line number and
                the second parameter specifies the column
                number.  If no parameter is given, the
                cursor is moved to the home position
                (Row 1, Column 1).
                  Example: ESC[10;20H  - moves the cursor to
                              row 10, column 20.

Cursor Up (CUU)
  ESC[#A            Moves the cursor up # lines without
                changing columns.  The value of #
                determines the number of lines to move up.
                This sequence is ignored if the cursor is
                already on the top line.
                  Example: ESC[5A       - moves the cursor up
                              5 lines without changing
                              columns.

Cursor Down (CUD)
  ESC[#B            Moves the cursor down # lines without changing
                columns.  The value of # determines the number
                of lines to move down.
                This sequence is ignored if the cursor is
                already on the bottom line.
                  Example: ESC[5B       - moves the cursor down
                              5 lines without changing
                              columns.

Cursor Forward (CUF)
  ESC[#C            Moves the cursor forward # columns without
                changing lines.  The value of # determines
                the number of columns moved forward.
                This sequence is ignored if the cursor is already
                in the rightmost column.
                  Example: ESC[25C      - moves the cursor forward
                              25 columns.

Cursor Backward (CUB)
  ESC[#n            Moves the cursor back # columns without changing
                lines.  The value # determines the number of
                columns moved backwards.
                This sequence is ignored if the cursor is already
                in the leftmost column.
                  Example: ESC[1n       - moves the cursor backwards
                              1 column.

Horizontal and Vertical Position (HVP)
  ESC[#;#f          This control sequence is the same as CUP.
                  Example: ESC[10;20f  - moves the cursor to
                              row 10, column 20.

Device Status Report (DSR)
  ESC[6n            Upon receipt of this command, the console
                driver will output a CPR sequence as described
                below.
```

```
Cursor Position Report (CPR)
  ESC[#;#R          The CPR sequence reports the current cursor
               position through the standard input device.  The
               first parameter specifies the current line and
               the second parameter specifies the current column.

Save Cursor Position (SCP)
  ESC[s             The current cursor position is saved.  This
               cursor position can be restored with the RCP
               sequence.

Restore Cursor Position (RCP)
  ESC[u             Restores the cursor to the value it had when
               the control sequence SCP was received.

Erase in Display (ED)
  ESC[2J            Erases all of the screen and the cursor goes
               to the home position (row 1, column 1).

Erase in Line (EL)
  ESC[k             Erases from the cursor to the end of the line
               and includes the cursor position.

Set Graphics Rendition (SGR)
  ESC[#;...;#m      Set the character attribute specified by then
               parameter(s). All following characters will
               have the attribute according to the parameter(s)
               until the next occurrence of SGR.
                 Note:  attribute means the foreground color, the
                     background color, blink, high intensity,
                     underscore, reverse video, and invisible.

              Parameter    Meaning
                 0    All Attributes Off (white on black)
                 1    Bold On (high intensity)
                 4    Underscore On (Some monitors only)
                 5    Blink On
                 7    Reverse Video
                 8    Cancelled On (invisible)
                 30   Black Foreground
                 31   Red Foreground
                 32   Green Foreground
                 33   Yellow Foreground
                 34   Blue Foreground
                 35   Magenta Foreground
                 36   Cyan Foreground
                 37   White Foreground
                 40   Black Background
                 41   Red Background
                 42   Green Background
                 43   Yellow Background
                 44   Blue Background
                 45   Magenta Background
                 46   Cyan Background
                 47   White Background

                 Example: ESC[33;40;1m    - all following
```

```
                              characters will have
                              a Yellow foreground,
                              a Black background,
                              and be in high intensity
                              until receipt of another
                              SGR control sequence.

                    ESC[0m         - all following
                              characters will have
                              a white foreground, on
                              a black background, in
                              normal intensity.

               Note:  Several parameters can be stacked.  For
                   example, ESC[0;1;5;7;31;44m
                         the above example will reset the
                         attributes, set high intensity,
                         set blink on, set reversed video,
                         set foreground color to red, and
                         set background color to blue.
                         Note that since reverse video
                         is on the foreground will
                         actually be blue and the
                         background will be red.
```

Here is the requested list of ANSI control sequences. I picked it up off of the Usenet a while back.

ANSI Standard (X3.64) Control Sequences for Video Terminals and Peripherals
in alphabetic order by mnemonic

(Inspired by the article "Toward Standardized Video Terminals: ANSI X3.64 Device Control" by Mark L. Siegel, April 1984 BYTE, page 365)

Note: This describes the VT-100 standard.

(Ps and Pn are parameters expressed in ASCII.)
(Numeric parameters are given in decimal radix.)
(Abbreviations are explained in detail at end.)
(Spaces used in this table for clarity are not used in the actual codes.)

Sequence Mnemonic	Sequence Name	Sequence	Default Parameter Value	Type or Mode
APC	Applicatn Program Command	Esc Fe		Delim
CBT	Cursor Backward Tab	Esc [Pn Z	1	EdF
CCH	Cancel Previous Character	Esc T		
CHA	Cursor Horzntal Absolute	Esc [Pn G	1	EdF
CHT	Cursor Horizontal Tab	Esc [Pn I	1	EdF
CNL	Cursor Next Line	Esc [Pn E	1	EdF
CPL	Cursor Preceding Line	Esc [Pn F	1	EdF
CPR	Cursor Position Report	Esc [Pn ; Pn R	1, 1	
CSI	Control Sequence Intro	Esc [		Intro
CTC	Cursor Tab Control	Esc [Ps W	0	EdF
CUB	Cursor Backward	Esc [Pn D	1	EdF
CUD	Cursor Down	Esc [Pn B	1	EdF

CUF	Cursor Forward	Esc [Pn C	1	EdF
CUP	Cursor Position	Esc [Pn ; Pn H	1, 1	EdF
CUU	Cursor Up	Esc [Pn A	1	EdF
CVT	Cursor Vertical Tab	Esc [Pn Y		EdF
DA	Device Attributes	Esc [Pn c	0	
DAQ	Define Area Qualification	Esc [Ps o	0	
DCH	Delete Character	Esc [Pn P	1	EdF
DCS	Device Control String	Esc P		Delim
DL	Delete Line	Esc [Pn M	1	EdF
DMI	Disable Manual Input	Esc \		Fs
DSR	Device Status Report	Esc [Ps n	0	
EA	Erase in Area	Esc [Ps O	0	EdF
ECH	Erase Character	Esc [Pn X	1	EdF
ED	Erase in Display	Esc [Ps J	0	EdF
EF	Erase in Field	Esc [Ps N	0	EdF
EL	Erase in Line	Esc [Ps K	0	EdF
EMI	Enable Manual Input	Esc b		Fs
EPA	End of Protected Area	Esc W		
ESA	End of Selected Area	Esc G		
FNT	Font Selection	Esc [Pn ; Pn Space D	0, 0	FE
GSM	Graphic Size Modify	Esc [Pn ; Pn Space B	100, 100	FE
GSS	Graphic Size Selection	Esc [Pn Space C	none	FE
HPA	Horz Position Absolute	Esc [Pn `	1	FE
HPR	Horz Position Relative	Esc [Pn a	1	FE
HTJ	Horz Tab w/Justification	Esc I		FE
HTS	Horizontal Tab Set	Esc H		FE
HVP	Horz & Vertical Position	Esc [Pn ; Pn f	1, 1	FE
ICH	Insert Character	Esc [Pn @	1	EdF
IL	Insert Line	Esc [Pn L	1	EdF
IND	Index	Esc D		FE
INT	Interrupt	Esc a		Fs
JFY	Justify	Esc [Ps ; ... ; Ps Space F	0	FE
MC	Media Copy	Esc [Ps i	0	
MW	Message Waiting	Esc U		
NEL	Next Line	Esc E		FE
NP	Next Page	Esc [Pn U	1	EdF
OSC	Operating System Command	Esc]		Delim
PLD	Partial Line Down	Esc K		FE
PLU	Partial Line Up	Esc L		FE
PM	Privacy Message	Esc ^		Delim
PP	Preceding Page	Esc [Pn V	1	EdF
PU1	Private Use 1	Esc Q		
PU2	Private Use 2	Esc R		
QUAD	Typographic Quadding	Esc [Ps Space H	0	FE
REP	Repeat Char or Control	Esc [Pn b	1	
RI	Reverse Index	Esc M		FE
RIS	Reset to Initial State	Esc c		Fs
RM	Reset Mode	Esc [Ps l	none	
SD	Scroll Down	Esc [Pn T	1	EdF
SEM	Select Edit Extent Mode	Esc [Ps Q	0	
SGR	Select Graphic Rendition	Esc [Ps m	0	FE
SL	Scroll Left	Esc [Pn Space @	1	EdF
SM	Select Mode	Esc [Ps h	none	
SPA	Start of Protected Area	Esc V		
SPI	Spacing Increment	Esc [Pn ; Pn Space G	none	FE
SR	Scroll Right	Esc [Pn Space A	1	EdF
SS2	Single Shift 2 (G2 set)	Esc N		Intro
SS3	Single Shift 3 (G3 set)	Esc O		Intro

```
SSA  Start of Selected Area    Esc F
ST   String Terminator         Esc \                              Delim
STS  Set Transmit State        Esc S
SU   Scroll Up                 Esc [ Pn S               1         EdF
TBC  Tab Clear                 Esc [ Ps g               0         FE
TSS  Thin Space Specification  Esc [ Pn Space E         none      FE
VPA  Vert Position Absolute    Esc [ Pn d               1         FE
VPR  Vert Position Relative    Esc [ Pn e               1         FE
VTS  Vertical Tabulation Set   Esc J                              FE
---------------------------------------------------------------------------
```

Abbreviations:

Intro	an Introducer of some kind of defined sequence; the normal 7-bit X3.64 Control Sequence Introducer is the two characters "Escape ["
Delim	a Delimiter
x/y	identifies a character by position in the ASCII table (column/row)
EdF	editor function (see explanation)
FE	format effector (see explanation)
F	is a Final character in an Escape sequence (F from 3/0 to 7/14 in the ASCII table) a control sequence (F from 4/0 to 7/14)
Gs	is a graphic character appearing in strings (Gs ranges from 2/0 to 7/14) in the ASCII table
Ce	is a control represented as a single bit combination in the C1 set of controls in an 8-bit character set
C0	the familiar set of 7-bit ASCII control characters
C1	roughly, the set of control characters available only in 8-bit systems. This is too complicated to explain fully here, so read Jim Fleming's article in the February 1983 BYTE, especially pages 214 through 224.
Fe	is a Final character of a 2-character Escape sequence that has an equivalent representation in an 8-bit environment as a Ce-type (Fe ranges from 4/0 to 5/15)
Fs	is a Final character of a 2-character Escape sequence that is standardized internationally with identical representation in 7-bit and 8-bit environments and is independent of the currently designated C0 and C1 control sets (Fs ranges from 6/0 to 7/14)
I	is an Intermediate character from 2/0 to 2/15 (inclusive) in the ASCII table
P	is a parameter character from 3/0 to 3/15 (inclusive) in the ASCII table
Pn	is a numeric parameter in a control sequence, a string of zero or more characters ranging from 3/0 to 3/9 in the ASCII table

Ps   is a variable number of selective parameters in a control sequence
       with each selective parameter separated from the other by the code
       3/11 (which usually represents a semicolon); Ps ranges from
       3/0 to 3/9 and includes 3/11

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v
v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v

Format Effectors versus Editor Functions

A format effector specifies how the final output is to be created.
An editor function allows you to modify the specification.

For instance, a format effector that moves the "active position" (the cursor or equivalent) one space to the left would be useful when you want to create an over strike, a compound character made of two standard characters overlaid. Control-H, the Backspace character, is actually supposed to be a format effector, so you can do this. But many systems use it in a nonstandard fashion, as an editor function, deleting the character to the left of the cursor and moving the cursor left. When Control-H is assumed to be an editor function, you cannot predict whether its use will create an over strike unless you also know whether the output device is in an "insert mode" or an "overwrite mode". When Control-H is used as a format effector, its effect can always be predicted. The familiar characters carriage return, linefeed, formfeed, etc., are defined as format effectors.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v
^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

ANSI X3.64 Mode-Changing Parameters for use with the
Select Mode (SM) and Reset Mode (RM) functions

Parameter Characters column/ row	graphic repres.	Mode Mnemonic	Mode Function
3/0	0		an error condition
3/1	1	GATM	guarded-area transfer mode
3/2	2	KAM	keyboard action mode
3/3	3	CRM	control representation mode
3/4	4	IRM	insertion/replacement mode
3/5	5	SRTM	status-reporting transfer mode
3/6	6	ERM	erasure mode
3/7	7	VEM	vertical editing mode
3/8	8		reserved for future standardization
3/9	9		reserved for future standardization
3/10	:		reserved separator for parameters
3/11	;		Standard separator for parameters
3/12	<		reserved for private (experimental) use
3/13	=		reserved for private (experimental) use
3/14	>		reserved for private (experimental) use
3/15	?		reserved for private (experimental) use
3/1 3/0	10	HEM	horizontal editing mode
3/1 3/1	11	PUM	positioning unit mode
3/1 3/2	12	SRM	send/receive mode
3/1 3/3	13	FEAM	format effector action mode
3/1 3/4	14	FETM	format effector transfer mode

```
3/1  3/5      15         MATM         multiple area transfer mode
3/1  3/6      16         TTM          transfer termination mode
3/1  3/7      17         SATM         selected area transfer mode
3/1  3/8      18         TSM          tabulation stop mode
3/1  3/9      19         EBM          editing boundary mode
3/1  3/10     1:                      reserved separator for parameters
3/1  3/11     1;                      Standard separator for parameters
3/1  3/12     1<                      error condition--unspecified recovery
3/1  3/13     1=                      error condition--unspecified recovery
3/1  3/14     1>                      error condition--unspecified recovery
3/1  3/15     1?                      error condition--unspecified recovery
3/2  3/0      20         LNM          linefeed/newline mode (not in ISO 6429)
3/2  3/1      21
 .            .
 .            .                       reserved for future standardization
 .            .
3/9  3/9      99

3/12 3/0      <0
 .            .
 .            .                       reserved for private (experimental) use
 .            .
3/15 3/15     ??
```

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v
^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

NOTES ON THE DEC VT100 IMPLEMENTATION

In the case of the popular DEC VT100 video-terminal implementation, the only mode that may be altered is the linefeed/newline (LNM) mode. Other modes are considered permanently set, reset, or not applicable as follows:

```
     Set:   ERM
     Reset: CRM, EBM, FEAM, FETM, IRM, KAM, PUM, SRTM, TSM
     N/A:   GATM, HEM, MATM, SATM, TTM, VEM
```

Control sequences implemented in the VT100 are as follows:

```
     CPR, CUB, CUD, CUF, CUP, CUU, DA, DSR, ED, EL, HTS, HVP, IND,
     LNM, NEL, RI, RIS, RM, SGR, SM, TBC
```

plus several private DEC commands.

Erasing parts of the display (EL and ED) in the VT100 is performed thus:

```
     Erase from cursor to end of line           Esc [ 0 K    or Esc [ K
     Erase from beginning of line to cursor     Esc [ 1 K
     Erase line containing cursor               Esc [ 2 K
     Erase from cursor to end of screen         Esc [ 0 J    or Esc [ J
     Erase from beginning of screen to cursor   Esc [ 1 J
     Erase entire screen                        Esc [ 2 J
```

The VT100 responds to receiving the DA (Device Attributes) control

```
     Esc [ c    (or Esc [ 0 c)
```

by transmitting the sequence

```
     Esc [ ? l ; Ps c
```

where Ps is a character that describes installed options.

The VT100's cursor location can be read with the DSR (Device Status Report) control

```
     Esc [ 6 n
```

The VT100 reports by transmitting the CPR sequence

```
     Esc [ Pl ; Pc R
```

where Pl is the line number and Pc is the column number (in decimal).

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v
^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

The complete document describing the standard, "ANSI X3.64-1979: Additional Controls for Use with the American National Standard Code for Information Interchange," can be ordered for $13.50 (plus $4 postage) from

Standards Sales Department
American National Standards Institute
1430 Broadway
New York, NY 10018
212/354-3300

It's best to read the full standard before using it. It also helps to have copies of the related standards "X3.4-1977: American National Standard Code for Information Interchange" (the ASCII standard) and "X3.41.1974: Code-Extension Techniques for Use with the 7-Bit Coded Character Set of American National Standard for Information Interchange."

See also the chapter "Using Extended Screens and Keyboard Control" in the IBM PC-DOS manuals (versions 2.0, 2.1, and 3.0), especially for the coding for character attributes.

The specification for the DEC VT100 is document EK-VT100-UG-003, available for $13.00 prepaid from:

Digital Equipment Corporation
Accessories and Supplies Group
POB CS-2008
Nashua, NH 03061

_O

```
==============================================================================
DOCUMENT apple.app
==============================================================================

Combining Applesoft with Assembly
Inspired by an article in Nibble
==================================
```

  The latewt issue of Nibble Magazine ,October '85, has a very interesting article in it regarding the combination of Applesoft and Assembly into one Applesoft file. I am extremely upset that I didn't think of this before! It really is an interesting trick! If you have read the article and had trouble using the instructions provide there or if you did not see the article, then please read on....

  When an Applesoft (referred to as A/S from here on) pvogram is loaded into memory either as you program or loaded from disk, the length of the A/S program is held in a specific memory location: AF.B0.   To add some additional assembly
to the program, you merely move the assembly code in memory down to the location right after whatever is stored in AF.B0. For example:

```
  1.  The A/S program is loaded first.
  2.  Enter monitor by using CALL -151.
  3.  Type AF.B0 anl read the last address used by the A/S program.
  4.  Remember that the lo-byte is used first (AF) and that the hi-byte (B0) is
      last.  If you see this:

     AF.B0-18 08

      this means that the last address used by the A/S program is 0818.
  5.  The SAVE command in A/S uses this location as a pointer to the last
      aldress in the program and saves everything from 0800 upward to the
      address held at AF.B0.
  6.  To add the assembly program to an A/S file, you would BLOAD the assembly
      portion just past the end of the A/S program.

     BLOAD ULTIMA CHEAT OBJ.,A$0819

  7.  Thys will put the assembly right on Top of the A/S program!
  8.  OK but how do I save it???  Easy!  Simply add the length of the assembly
      program to the address at AF.B0 anl the change theaddress at AF.B0 to
      reflect the longer length.

     0818   Address found at AF.B0
    +2200   Length of the assembly pgm.
    -----
     2A18   Address to put at AF.B0!

     CALL -151warm entry to A/S)

   ]SAVE ROUTINE
```

```
=======================================
PROBLEMS AND DIFFICULTIES THAT HAPPEN
=======================================
```

Of course, it is necessary to do one of two things in order to make aall this work. Either make your assembly code relocatable (which you should do in any case) or include a memory move to relocate t`g assembly at the proper address once it has loaded. Here is a fool proof memory move in A/S:

```
  10 POKE 66,[DESTINATION-LO]:POKE 67,[DESTINATION-HI]
  20 POKE 60,[START-LO]:  POKE 61, [START-HI]
  30 POKE 62,[END-LO]:  POKE 63,[END- HI]
  40 POKE 71,0 [SET THE Y-REGISTER TO 0.  IT IS USED AS AN INDEX BY THE MOVE
     ROUTINE THAT WE WILL CALL NEXT]
  50 POKE 58,44:  POKE 59,2%4 [THIS ESTABLISHES THE MOVE ROUTINE POINTERS]
  60 CALL -327 [THIS IS THE ACTUAL MOVE ROUTINE]
```

If you do not understand the above, simply use it as is and everything will work find. Of course, you will have to know the addresses but that is done as explained above.

One problem that you WILL encounter is related to the HIRES PAGE 1. Lets say that you have a program that you want to run and then it is to display a hires picture on HIRES 1. Since our A/S programs start at 0800 and a hires picture length is $1FF8 or as is more popularily accepted (DAMMIT IT IS WRONG AND YOU WASTE ONE WHOLE SECTOR OF DISK SPACE DOING THAT) $2000, if you add $0800+$2000 the answer is $2800! Since HIRES 1 starts at $2000 you will obviously be overwritting HIRES 1 and the results is GARBAGE! In a case like this, go ahead and waste some disk space and change AF.B0 to $3FF8 so that when you save the A/S program you are saving the entire HIRES 1 along with it.

======================================

If you have problems doing this, I will be delighted to assist! Leave me a message on the Rebel Alliance Pro System (leave it in feedback and Tracker will get it to me) at...

       206-584-6900

This is a 10meg no hangup after one download kixass system so call it anyway!!!

See you Later.... Hoe Hopper
=======================================

```
=============================================================================
DOCUMENT apple.txt
=============================================================================
```

NEW TECHNOLOGY ANNOUNCEMENT FROM APPLE

In a surprise announcement yesterday, Apple Computer said that is is finally doing away with the keyboard. Apple stated that the microcomputer user has suffered too long with this awkward and inefficient input device. According to an Apple spokesperson, the technology for replacing the keyboard with only a mouse is here and the computer user is ready for it. The spokesperson said that Apple has received a steady stream of complaints over the years about the need to constantly move the hands between the Mac keyboard and the mouse. "The solution was obvious - do away with the keyboard completely."

Acknowledging that there are still a few Mac applications that depend on textual input in addition to graphical manipulation, Apple said the poor people stuck with such outdated technology have not been forgotten. They are introducing the Spinning Alphabet Wheel (SAW) to replace the keyboard. The SAW is a screen display object consisting of concentric circular strips showing all of the characters which normally appear on the keyboard. The wheel rotates continuously under character selector windows. The user selects a character by placing the mouse pointer in the appropriate window at the same time as the desired character is about to appear. "...and, ta-da, the selected character appears on the screen just as though it had been typed on an old fashioned keyboard."

"This is a marvelous new technology with plenty of room for growth," said the spokesperson. For example, the user can configure separate wheels for vowels vs. consonants. Or digits can be placed on their own special low speed wheel. "We have conceptualized the keyboard as a big, bulky menu selection device and replaced it with dynamic display menus instead. Apple will eventually replace all menus with their new Rotating Wheel Technology (RWT)."

When asked why the wheels have to rotate, the spokesperson said that Apple's engineers had considered using conventional "point-and-click" technology for the wheel. "However, we feel that this type of operation is too complicated for the typical Mac user. So, we have done away with the mouse button, too. It is still hard for us to believe that the IBM world has stepped backwards in technology by providing two or more buttons to confuse the user. The IBM compatible sector has not yet recognized that 95% of computer usage is devoted to experimenting with different fonts and character styles in documents."

Asked if this new technology would reduce the price of the typical Mac computer, the spokesperson countered that it would probably increase the price of the Mac. "After all, display space is already scarce on the current screen. We will now deliver Macs with two screens - one for the normal display and a larger one for the multitude of rotating wheels the user needs to access." Apple said that the user who is confused by complicated devices such as keyboards and mouse buttons will gladly pay a premium to avoid them. "In fact, the easily-confused user is our best customer," replied the spokesperson. "Not only are we doing away with the pesky keyboard, but we are also giving them something they have demanded for a long time - more screen space. this is definitely a win-win situation."

Beta testers of the new technology were impressed by its ease of use, but said there are still some minor problems to work out. for example, one tester left his machine unattended with the uppercase character wheel spinning at medium speed. While he was away somebody must have jarred his desk, moving the mouse pointer into the selector window. When he got back he found that his Word document now had one huge paragraph consisting of all of the characters of the uppercase alphabet repeated 2,539,987 times. "At first glance, this appeared to be a big problem. But after I formated the new paragraph with 33 different fonts and 11 different type styles, it looked great. I hope that Apple fixes this problem before they release it, because these accidents can greatly increase the time spent formatting documents."

```
=============================================================================
DOCUMENT apple2.gs
=============================================================================

Path: bimacs!barilvm!psuvm!psuvax1!rutgers!mailrus!uunet!looking!funny-request
From: rww@demon.siemens.com (Richard W West)
Newsgroups: rec.humor.funny
Subject: Apple IIgs
Keywords: computer, original, chuckle
Message-ID: <S129.451b@looking.on.ca>
Date: 10 Jun 90 23:30:06 GMT
Organization: Crazy Productions
Lines: 198
Approved: funny@looking.on.ca
```

{ed Well, what can I say. This piece is long and rather nasty to Apple, but it has its good moments. If you're from Apple, don't read it.}

Ever watch a TV show and someone'll say:

"Take this, you son-of-a-[BEEP]!!"?

And you, of course, fill in the missing word.

"Son-of-a-Bitch," you say in your mind.

And so does every human being who hears the beep, because the mind naturally completes recognized patterns, no matter how fragmented they are. We know that "bitch" follows "son of a" just like night follows day or Pete Rose follows bookies. Unless of course you're Russian. Then, "son of" is usually followed by "Ivan" or "Mikhail" or somesuch male Russian name that reminds you of various blackspots on the otherwise clean and white tapestry of Russian history.

But anyways, I was talking about how censors cover-up swear words with bleeps and stupid sound effects. The whole reason they go through all this bullshit and cut up decent movies into nonsequential nonsense is so our children won't hear words usually only associated with the description of Apple products.

Now, I don't wanna get on anyone's case about this, but I am wholly amazed by Apple's IIGS system. This is the most mazing case of reverse technology in computer history! In the age of the 386/33 and the 486/25, Apple Computer comes out with a machine that runs at an astounding 2 mHz! AND, you can speed it up to a blinding 2.5 mHz. HOLY SHIT! Two point five? You know how they play THAT one off? The salespeople tell ya "Yeah, and the high-speed mode speeds up the CPU 25 PERCENT!" What the HELL is this?! Did some dude at Apple get Woz really plastered and then say:

"Hey Steve! I'll tell ya what -- if we can't beat 'em at making the fastest machine, by God -- we'll beat 'em at making the SLOWEST!"?

Because they DID!

This thing's like a slug in winter! And as if this Yugo CPU wasn't bad enough, they get a disk drive straight out of computer Hell! You ever load a program on a Commodore 64? I mean waiting 20 minutes for Zork to load may have seemed like a long time then, but you load Zork on a IIGS, and boy, you are in for a WAIT! Your grandkids'll be sittin' there waitin' for that fucker to load. I'm not joking! If you have a monitoring program, you can see the drive plinking off bits in a completely leisurely manner. Plodding doesn't even BEGIN to describe it! Plodding suggests MOVEMENT, and if can master the almost Zen Buddhist-like art of sitting still in front of a II GS drive long enough to detect the motion of the disk, then you've got pretty good idea of what taking THORAZINE is like! If you can sit still that long, you're qualified to be a National Monument!

And when you pay for it, it's like you bought a C-64, BUT AT AN IBM PRICE! A good (and that's a word not commonly used in conjuncture with "GS") system will put you back close to $3,000! You can get a decent (there's another one of those aforementioned words) 286 system for that! And a 286/12 kicks this thing's ass so many ways you can't count 'em! (at least not on a GS).

They call it "GS", but they don't tell you it stands for "Goddamn SLOW"!

Oh, it's got great graphics. Serious, this thing's got the graphics. But it's like having unlimited credit at a Goodwill shop! You'll also wait MONTHS before their demo picture of the golden King Tut finally gets to the screen. The whole time, at wholly random intervals, you'll get messages like:

"Now computing byte 53, bit 6, of 648,457 bytes. Next report in 20 mkNf."

And just about the time you've gotten the .12 gauge outta the closet with the computracide on your mind, the little fucker'll pop up some Fable ROM program imbedded in its enfeebled memory and tell you a little a story like:

"Once upon a time, there was a sloth and a cheetah. The cheetah was a very fast cat and the sloth a plodding oaf. Too many times was the cheetah caught speeding by the CHP (oh hey, for those of you who have the unfortunate fate of living outside California, CHP means California Highway Patrol) and the CHP sawed the poor cheetah's legs off.

"Moral: People who buy fast machines often get their legs sawn off."

You stand there, looking dumbfounded.

"What?" you ask yourself.

And the GS answers:

"Now computing byte 53, bit 7, of 648,457 bytes. Next report in 20mins."

You'll blast that son of a ----- (y'all said 'bitch', dinja? See? straight to computer Hell where some poor bastard'll hafta wait for it to compute PI to 20 billion digits (NOW you know why you need PI calculate billion digits!) before he can go to Heaven!

THIS THING IS A TURD! It's a cattle-dropping of a computer! The day Apple introduced this li'l gem was forever to be known as "The Day

Silicon Valley Smelled like BULLSHIT" because if you call a turd "a rose", IT'S STILL A TURD! I don't care what PR says! If it's brown and smells like shit, IT'S PROBABLY A TURD!

You want to have some fun with an Apple Dealer? Get dressed in your best business suit and walk into AppleLand or any store that only sells Apple and go up to a dealer. Look for the slimiest one. Tell him you've been thinking about getting a II GS for your family for months and have finally decided to buy the best II GS system available. Now if you could only see a demo, you would be convinced that you were spending your money wisely. They'll put on the dog and pony show for ya and show you some cute program.

Then tell the dealer you use "Harvard Graphics" a lot at work (or any other HUGE program they have II GS versions of) and seeing how it looks on the II GS would close the deal. And the whole time, talk about hard drives, expensive monitors, and lots of software. But don't over-do it or they'll figure you out.

But anyway, when he plops in the full 1.8 meg floppy to be read at 300 baud by the disk drive, you start a conversation, and casually introduce how speed in a computer is important to you. Mention the fact that you work with a 386/33 at your office and tell him that the baby really flies! Keep talking about how impressed you are with the 22 millisecond access time on a Compaq 110 megabyte hard drive. Tell 'em how you load MS Windows in 3 seconds. 2 seconds for Harvard Graphics. Then, VERY casually look at the II GS drive, then look at your watch. Frown. Do it again with a very slight look of disbelief. Ask, "Is it done yet?" quizzically . . .

Watch the little weasel SWEAT!

Oh, it will do you a WORLD of good!

Caution! If you start busting up now, it's OVER! You won't EVEN stop! But if he recovers and gets the conversation going again, just look over towards the drive every so often and sound slightly more irritated each time you reply to the dealer. THEN: Look at the drive and then at your watch again. Look the dealer right in the eye and ask, "Is it done loading yet?" with a little more irritation.

Just see how many times you can repeat the cycle. When the Dealer starts getting really pissed about being asked "Is it done loading yet?" a million times and gets rude, or the program actually loads, you close the deal. But then look at the computer, then at your watch and tell him "I need to reconsider this. I'll come by again if I decide to stay with this machine." Look at the machine and shake your head while saying "But it's doubtful," and walk out.

You will have fucked that guy's day!

If you wanna really dangle the dude on the hook, get him to admit Compaq is superior to Apple. Just mention the 386/33 in your "office" again and then ask:

"What does this machine run at? 10 megahertz? 8?"

"2" the Dealer will admit.

Look him right in the eye and ask in the your most astounded voice:

"2?"

The dealer will shrivel up like a snail with salt poured on it!

If you have enough Apple-only dealers in your town, you and a friend can make an afternoon of it! And every time you walk out of one of thos places, after rightfully humbling those toadies, you feel at one with nature, and animals will cross the street to be near you.

Heh heh heh.

I'm sorry, before I got off on all the Apple stuff, (by the way, I don't want everyone thinking I hate Apple computers. That was just a little good-natured prod to remind the folks at Apple which half of the 80's we are in.) I wuz talkin' about censoring TV programs for the sake of our chidrens' language. Okay, do this:

Tonight when you go home, walk up to your kid and ask him/her to complete this sentence:

"Mike Tyson is one bad mother _______!"

Your kid'll look you straight in the eye and say:

"Fuhka."

This is the exact and true nature of what censoring TV movies accomplishes.

```
=============================================================================
DOCUMENT appleii.jok
=============================================================================
```

Article 181 of rec.humor.funny:
>From: flong@watmath.UUCP (Fred J. E. Long)
Subject: My Apple][test
Date: 30 Nov 89 11:30:07 GMT

Unofficial APPLE][Brainwash Test

Have you been brainwashed by your past experience as an Apple][hacker? Here is a test you can take to find out.

1. What is /r$?
    a) "slash r string"
    b) "slash r dollar sign"
    c) a subdirectory of the root directory
    d) Rich Salz

2. Do people wonder why you keep using "Applesoft" as a synonym for BASIC?

3. Do you despise assemblers, prefering instead to code your programs byte by byte in machine language with a debugger?

4. Do you only use three registers when programming because "if A, X, and Y are good enough for the 6502, then by golly they're good enough for me"?

5. Do you still have floppies that have write-protect holes on both sides, but are labeled "single sided"?

6. Are you uncomfortable with the words "interrupt," "timer," or "multitasking"?

7. Do you have "Beneath Apple DOS"?

8. Do you wonder why any Gentleman would need more than 64K?

9. Are you distrustful of lowercase?

10. Do you have "alias CATALOG ls" in you .login?

11. Do you despise anything that is not overtly user-hostile?

12. Wonder why & doesn't do the same thing in UNIX?

13. Think ^D in UNIX is a DOS command?

```
=============================================================================
DOCUMENT applemaf.txt
=============================================================================

******************************************************************************
*******                                                                *******
*******                      The Apple Mafia Story                     *******
*******                                                                *******
*******                      As Told To: Red Ghost                     *******
*******                                                                *******
******************************************************************************
```

The reason I'm writing this file is to (hopefully) once and for all, clear up all the rumors, false statements, and just lies, that are going around about one of the first 3 pirate groups ever. (Untouchables, Apple Mafia, Dirty Dozen)

I won't get into the other two, because there already is "The Untouchables" story, and I'm not qualified to make comments about the Dirty Dozen.

Everything in this file is fact. I grew up in Queens NY (now 718). Where many of the "original" pirates and phreaks, were from. Many of the readers will question certain aspects of this file, or my authority to write it. But they are always more than welcome to go to the sources themselves and find the truth is what I am writing.

****

To begin with, I'm now 22 and going to college. I have been out of the "wares world" for the better part of 2 years. What prompted me to write this is a younger friend of mine, who is now a "pirate" and spends his life calling all the boards, and getting all the wares (Not making fun of anyone, I went through the same phase when I started, but it was very different from the pirates world of today). About a week ago he told me about the "Apple Mafia" regrouping. I said bullshit. And I was right. He gave me some of the files on discussions now going on about the "new" "apple mafia" and I must say it's pretty sad.

I felt the real story of what went on should be told, so here it is.

One more thing before I start. I was never in the Apple Mafia, or any of the other groups mentioned here. I never had that deep an interest in computers as anything but game machines or better typewriters. But I grew up in close proximety to many of the people involved, and spent time with them in other areas besides computers.

****

From the messages posted about the controversy I have seen. I think it might be best to go through them 1 by 1. So here's the first one:

(Buffer of msg's. Untouched by me, except the conversion from 40 to 80 cols.)

****

To:The Fake Apple Mafia   From:Disk Rigger

You are definatly not the real Apple Mafia. It was a popular group a while ago.

It includted Bioc Agent 003, Tuc, Hight Technology, Creative Cracker and a few others. I will inform them of you losers using there names and you will be ragged out of your minds. I would recomend you changed your group name or you will feel the wrath of Bioc!
This is serious and you losers can keep your unoriginal name but You will pay for it.

(Someone's reply.)

Disk Digger:
Get a life dude. The Apple Mafia is OVER. BIOC Agent was thrown out of the Apple Mafia. The final members were Tuc, Lord Digital, Creative Cracker, High Technology, Big Brother the Phantom, Silicon Scorpion, & Sherlock Apple. Nobody is going to Feel the wrath of Bioc (hahahahahhahahahahha). because 1. he quit the phreak/pirate world over a year ago. 2. if he WAS around he wouldnt give a shit. 3. The old members who are still around probably care even less.

(Someone elses reply.)

While the new "apple mafia" (get a life you fuckups. you might as well call yourselves the Untouchables.) are idiots.
new warez r0dentz who sure as hell weren't around when the REAL apple mafia was around, are even bigger r0dents. This means YOU disk digger.
Naming yourself after a program used to pack new wherez. get a life you fat fuckup. And if the people in the apple mafia knew YOU were throwing their names around on a fucking catfur, YOU are the one they would be mad at.

FUckup

****

Despite incorrect statements in both parties messeges, this seems like a good time to point out that Bioc Agent was never in the Apple Mafia to begin with. The members changed through time, but Bioc was never a member. (More on him at a later time) The Final members were: (If you doubt this, then boot up any ware cracked by them in 84-85. Which was the final regrouping of the Mafia)

The Godfather (Charles). The Phantom (Eric). Creative Cracker (Marc).
High Technology (Craig). Lord Digital (Patrick). Jacque Lafitte (Jack).
The Magnet (Joe). Sherlock Apple (Dave). Silicon Scorpion (Mark).
Data Dragon (Jordon). Big Brother (Andy). & towards the very end: Tuc (Scott).

Former members who were thrown out or quit: The Parasite, Tylenol Cyanide, Yosemite Sam, and people I don't remember from CA.

(More Messages)

****

```
Number > 7
Subject> Apple Mafia? (Ha)
Viewer > DISK RIGGER            <Elite>
Posted > MON MAR 17  4:24:17 PM
```

Genius. Are you talking about that
loser group in 213? They are in for it.

If you didnt know already there was
already a group Called Apple Mafia. Too
bad two of 'em were FEDS.

Hey, Where ANY of you ever on The Old
Sherwood Forest's? [/, //, ///]?

(Another msg.)

Number > 8
Subject> SHERWOOD FORESTS
Author > CHRONOS
Posted > MON MAR 17 5:03:32 PM

Yeah, I was on // and ///... I never got an answer at / though... Wasn't the
Apple Mafia (later Micro Mafia) running those boards?

[chronos/KOTBC]

And didn't they break up before they got busted? I remember something about
them just giving up, cause 2 guys were doing all the work...then they put
Yellowbeard in charge and it was never heard from again...

(Another msg.)

Number > 13
Subject> Sherwood Forest
Viewer > WARE BANDIT   <Elite>
Posted > TUE MAR 18 3:43:11 PM

What two members of The Real Apple
Mafia were feds? Because I knew several
including High tech, and Creative.

And the reason you neve got an answer
at sherwood forest / was because that
was the old sherwood forest //... High
tech moved from Ny, and after he set it
up at his new place he called it
Sherwood forest //.... Well I have many
old files, and posts saved from the
last grand adventure (that was thier
logo), So if you would like to view
them then just leave me a message on
the underground.

later, [Ware Bandit]

(Another msg.)

Number > 14
Subject> WB..
Viewer > DISK RIGGER   <Elite>
Posted > TUE MAR 18 4:17:52 PM

I think Magnetic Surfer ran it. Yeah I
also have a lot of things saved on
Buffer. They had a LARGE G-File
section. And a TRUELY awesome Elite
sub.

And some people got the impression that
Creative Cracker was a lazy ass. It is
not true. The board was being runned by
Federal Agents. He just came in to look
at it.

(Another msg.)

```
Number > 15
Subject> SERIOUS?
Author > CHRONOS
Posted > TUE MAR 18  5:44:17 PM
```

Sherwood Forests were really being run by Feds? I'd like to know who... they
let it get pretty far before they busted them then...

[chronos/KOTBC]

(Another msg.)

```
Number > 18
Subject> The forest
Viewer > WARE BANDIT            <Elite>
Posted > WED MAR 19  3:23:11 PM
```

Okay this might clear some things up between some people that might be confused
on the subject.

```
Sherwood Forest .... Used to be run by High Tech. About 2-3 years ago.
Sherwood Forest //   The later Sherwood forest run by High tech. when he moved.
Sherwood Forest ///  Ran by Creative Cracker, a good friend of High tech.
```

and no they were not feds, they just got a little carried away with the way
they were running thier boards by letting just about anyone access most of the
phreak sections that they had on-line. The Treasury dept. finally closed them
down along with cryton and some other un-heard of hack boards.

If you have any questions feel free to ask.. I was going to run thier number 4
board, but shortly after I was offered the opurtunity they went under...

****

I don't know where these msg's are from, or who these people are. But almost
everything said in them is wrong.

"I heard 2 guys were doing all the work, so they quit". Whoever said that has
obviously read the 1984 loserlist, because that is a verbatim quote from it.
Not only is it wrong in slandering everyone from Apple Bandit, Hot Rod, and
Wombat/Gonif, to Bioc Agent, Lord Digital and Paul Muad'Dib. But it's written

in a such a ridiculous fashion, that any points the author was trying to make are lost in the jumble (For those who care it was written by "The Atom").

Sherwood Forests (The last grand adventure).

The FIRST Sherwood Forest was started in 1979 by Magnetic Surfer. The only people who were on it back then, and still recognized would be Mr. Xerox and Lord Digital. Other members included Nickie Halflinger, Captain Crunch, Napoleon Bonaparte, and many others, whose names are probably meaningless to the people around today.

There was a 20 page write up on "Hackers", before it was in vogue to write about them (Before WarGames and the 414 busts), In a 1982 issue of "California" magazine. Which detailed the bust of Ron Austin, who was at the time of arrest 22 years old and being tried for everything from credit card fraud, to grand theft. Also included in the article was a write up on Hacker bbs systems he was on and people he knew. The Sherwood Forest mentioned in that article was Magnetic Surfer's, not the later ones.

Sherwood forest was run on a micromodem and 1 disk drive. There were no Cat's, DOS was at 3.1, and Disk Drives were still a novelty. It went down in 1981 for whatever reasons, and went back up as a public system with 1200 & a 46 mb hard drive in 1983. Shortly thereafter it went private and became the Knights of Shadow Base bbs. During it's final days in 1984, this was the FINAL memberlist:

SYSTEM OPERATOR	<-- Magnetic Surfer. TKOS member
PIT FIEND	<-- Local Queens person
LORD DIGITAL	<-- Name speaks for itself. Apple Mafia & TKOS member
CRIMINAL ELEMENT	<-- Local Queens person, semi notorious for being a ass
STEPHEN FALKEN	<-- aka: Jon Gleich of Earth News Central
THE SURGE	<-- Unknown. Access from Lord Digital
BIG BROTHER	<-- 617 phreak/pirate. Apple Mafia & TKOS member
E.F. HUTTON	<-- 312 Phreak/Hacker. TKOS member
THE KNIGHTS OF SHADOW	<-- TKOS account
MR. XEROX	<-- Name speaks for itself. TKOS founder
CAPTAIN AVATAR	<-- aka: Skip Rooney
DISK DEMON	<-- Local Queens person
GUEST ACCOUNT	<-- A guest account
THE MAGNET	<-- Apple Mafia co-founder
THE PROWLER	<-- Canadian Phreak, NOT the 612 Prowler
BIOC AGENT 003	<-- Name speaks for itself. TKOS member
QUASI MOTO	<-- TKOS member. Ran PloverNet
PHONE FIEND	<-- Local Queens person
TUC TUCBBS	<-- Tuc. Name speaks for itself. TKOS member
PAUL MUAD'DIB	<-- Name speaks for itself. TKOS member
HARDWARE HACKER	<-- Local Queens person
TOM TONE	<-- Brooklyn person
THE GODFATHER	<-- Apple Mafia co-founder
PETER MCIVERS	<-- 617 phreak/hacker
THE PHANTOM	<-- Apple Mafia member
NICKIE HALFLINGER	<-- Hacker, EEE, in his mid 30's
LEX LUTHOR	<-- 305 person. Nobody at this time, later founded LOD
UNCLE JOE	<-- Have no idea who this is
LESLIE KARASIC	<-- Have no idea who this is
THE NECROMANCER	<-- TKOS member
DR JIMMY MR JIM	<-- TKOS member
THE WIZARD	<-- Asshole from 713. Access from Quasi Moto
FRODO HOBBIT	<-- Unknown, from 201. Access from Magnetic Surfer

THE DJIN	<-- Unknown, from 718. Access from Magnetic Surfer
RICH DOUGAN	<-- Unknown, from 718. Access from Magnetic Surfer
APPLE CAT	<-- Unknown, from 718. Access from The Phantom
NAPOLEON BONAPARTE	<-- TKOS member. Ex of: Inner Circle. ex Sys of SL
CHRISTOPHER BUNN	<-- Unknown, from 718. Access from The Phantom
DR. DOOM	<-- Have no idea who this is
WILD CAT	<-- Have no idea who this is
THE DISKLAIMER	<-- Have no idea who this is
MR IBM	<-- Have no idea who this is
DRAGON LADY	<-- Ex girlfriend of Chesire Catalyst
GAP DRAGON	<-- Have no idea who this is
THE MARK	<-- Have no idea who this is
CABLE PAIR	<-- Fed par excellance`
STOSH FIXER	<-- Have no idea who this is
MILO PHONBIL	<-- Sysop of Once & Future OSUNY
MR. GUCCI	<-- Sysop of AT&T Phone Center
DR. NIBBLEMASTER	<-- TKOS member
STAINLESS STEAL RAT	<-- TKOS member

And that's it. Reading the list you have to keep in mind that this was right after the Inner Circle folded, and almost a year before LOD even began. So many of the current "Big Names", didn't even own computers yet.

In 1983 a friend of their's (The 212/718 people), called Creative Cracker put up a bbs called Sherwood Forest][. For a while it had an ae line simply known as "Sherwood Forest" in 201, then in 1984 Sherwood Forest]I[went up, run by High Technology. The original co-sysops on SF][were High Tech & Jack The Ripper. Jack was dropped, High Tech became a full sysop, and the new co-sysops became: Tuc, Bioc & Big Brother.

SF][started on a disk][& a Rana]I[disk drive, Micromodem, and a clock. It went down for 2 weeks in 1984, then came back up with all new software, 20 megs and 1000's of files.

SF]I[went up with a disk][& a Rana]I[, and that's as big as it ever got. Creative Cracker was a full sysop. And the co-sysops were Sharp Razor, X-Man, and Wizard 414. co-sub-ops of maintenence (whatever that is), were: Sherlock Apple and Silicon Scorpion. All co-sysops and sub-ops were dropped after about 2 months.

None of the boards except for Magnetic Surfer's could be considered really impressive by todays standards. Impressive in terms of hardware that is. This is the time when just about the only hard drive was a corvus, whose prices for a 6 meg started at $2000 in that time. and a complete Apple-Cat/212 card/expansion module system, came to over $1000.

And finally the Apple Mafia, was NEVER the "Micro Mafia", this was yet another group of losers cashing in on their group name.

****

I could go on about what happens to all the people from SF, but that's not the purpose of this file, so I'm not going to digress. There are many files out about many of the members (Lord Digital, Paul Muad'Dib, Mr. Xerox come to mind) already. And others have made themselves known through LOD as well (Lex). What this file is about is The Apple Mafia, so I'm not going to get into all that.

Which brings me to the final part. The Sherwood Forest Busts....

I'm saying right now that I don't know what happened, just offering the facts that I do know about it.

If (as is said), SF][was indeed run by the FBI towards the end. Then why did they let Creative Cracker keep cracking software? It doesn't seem likely they would have let him continue, did they just not know or what?

Now if they weren't running SF][, then even more questions come to mind. The most important one is: why run a board if you truly don't care? CC never looked at his board, never logged in, the news was updated once every 9 months by BIOC.

What's more, if it WAS a trap, then logically the other Apple Mafia members had to know about it. Or most of them. NONE of them EVER posted anything at all. With the exception of Silicon Scorpion & Sherlock Apple, who kept posting new wares.

It could be that they lost interest in piracy (As is true of Tuc, Lord Digital, The Phantom, Big Brother, in fact, almost all the members), BUT this still does not clear up why NONE of them EVER posted ANYWHERE, including the phreak boards or the elite subs. Now BIOC did post, but if you know BIOC, he always left some 500 line exerpt he typed up from some magazine, or manual (Which is actually just about all he ever did, much like Lex), nothing that could get him in any kind of trouble.

These are the same people whose msg's I've seen buffers of from Sherwood Forest (Magnetic Surfer's), and World of Cryton, who were posting techniques, systems, and information, all over. But not ONE msg. from any of them on their supposed "home base". Pretty weird....

The logical conclusion I draw from this is: They knew the board was being watched at the very least, and didn't want to draw attention to themselves. In which case they left everyone else to get caught, which is in keeping with many of the peoples present attitudes. This is also the time during which the Apple Mafia members who were also in TKOS got a lot of heat from that group falling apart in a rather spectacular manner. With almost all of them in danger of being busted for grand theft.

While SF][&]I[didn't go down until summer of 1985, I would say the Apple Mafia died almost a year eairlier. I talked to Silicon Scorpion towards the end of 1984 and as he put it: "What group? I don't even have anyones number anymore. How am I supposed to be part of a group whose members I can't even find?" This refers to Creative Cracker whose voice number dissapeared, and who never answered his feedback, The Phantom, ditto. Big Brother, ditto. Tuc, ditto. Lord Digital, ditto + he stopped calling any boards. Data Dragon, he's never home. Who's that leave? not many.

In the cases of The Phantom, Lord Digital and Tuc, this is understandable. They were all undoubtedly living out paranoid fantasies of everything they'd ever done catching up with them, as it did with Mr. Xerox not too far in the past.

But what happened to everyone else?

****

The Apple Mafia is over. Maybe one day some of the members will regroup just for fun. It would be interesting. But some loser named Judge Dredd, should go crawl back where he came from. Or find another name for his lame excuse of a group.

****

Here is the "flyer" heralding the regrouping of the Mafia in 1984. Verbatim as typed by The Godfather:

```
BRIEF HISTORY OF THE APPLE MAFIA.

FOUNDED IN 1980 BY THE GODFATHER AS
A JOKE.  REDONE IN 1981 AS A SEMI
SERIOUS GROUP.  KICKED SOME ASS IN
'82.  BLEW EVERYONE AWAY IN 83, AND
WILL DO MUCH BETTER IN 84.  SINCE
THE BEGINNING THE GROUP HAS DIED
OUT AND BEEN REBORN SEVERAL TIMES,
THIS TIME LETS KEEP IT GOING.  IS
CURRENTLY THE OLDEST ACTIVE GROUP,
NEXT (OF PEOPLE WHO WOULD STILL BE
AROUND) ARE THE WARE LORDS ('83 I
BEILIEVE) AND THE 1200 CLUB ('83
ALSO, I THINK).  THAT'S IT.

A FEW GENERAL IDEAS WE WISH TO PROMOTE:

WHEN YOU GET SOME NEW SOFTWARE CALL
UP AS MANY MAFIA MEMBERS AND GET IT
TO THEM FIRST.

GET IT POSTED UP ON THE BOARDS AS FAST
AS POSSIBLE, NOW WITH CATS, TIME IS
CRUTIAL.  BE SURE TO STICK THE MAFIA
AT THE END.

DON'T GET THE GROUP INTO FIGHTS, AND
TRY NOT TO BE AN ASSHOLE.

IT HELPS TO GIVE OUT SOFTWARE TO OTHERS
- WITH THE MOB BEHIND YOU, NO ONE
CAN STOP YOU.

KEEP IN TOUCH WITH OTHER MEMBERS, IT
SUCKS WHEN YOU LOSE CONTACT, IF YOU
NEED ANY HELP CATCHING UP, JUST ASK
SOMEONE... THEY SHOULD BE WILLING TO
HELP YOU CATCH UP.

NEW MEMBERSHIPS ARE DECEIDED BY ALL
OF THE MEMBERS, IF YOU KNOW OF SOMEONE
DECENT, TELL EVERYONE ELSE.

THATS ABOUT IT.

THE GODFATHER

        1986.
```

The Apple Mafia Story

```
      Red Ghost

*******************************************************************************
(>==============================================================================

       ___  ___
      (___><___)  .....And if you enjoyed this TextFile, call The Works, 914's
\        _[]_          TextFile BBS:
 \______/    \_____                           The Works
                  _\                          =========
  _______________ \>\     (914)-238-8195 24 Hrs.       900+ Textfiles Online
 /               \   \    300/1200 Baud, N,8,1         Home of Terror Ferret
/                 \___>   10 Megabytes of Storage      ANSI Graphics Optional
==============================================================================
Call The Works BBS - 1600+ Textfiles! - [914]/238-8195 - 300/1200 - Always Open


X-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-X

 Another file downloaded from:                               NIRVANAnet(tm)

 & the Temple of the Screaming Electron   Jeff Hunter          510-935-5845
 Rat Head                                 Ratsnatcher          510-524-3649
 Burn This Flag                           Zardoz               408-363-9766
 realitycheck                             Poindexter Fortran   415-567-7043
 Lies Unlimited                           Mick Freen           415-583-4102

   Specializing in conversations, obscure information, high explosives,
       arcane knowledge, political extremism, diversive sexuality,
       insane speculation, and wild rumours. ALL-TEXT BBS SYSTEMS.

  Full access for first-time callers.  We don't want to know who you are,
   where you live, or what your phone number is. We are not Big Brother.

                          "Raw Data for Raw Nerves"

X-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-X
```

==
DOCUMENT applenet.app
==

2

Apple-Net BBS Software
======================

<< Latest version of Apple-Net is 2.3F. Please note that pirated copies that you may have seen going around are NOT guaranteed to work. Only the latest version, purchased directly from us, is guarenteed to work as follows! >>

Apple-Net will run on any Apple //+, //e or //c with at least one disk drive and a modem. Apple-Net supports the Novation Apple Cat (at 300/1200 baud), Hayes Micromodem II and IIe, and the Apple Super Serial Card w/Smartmodem compatibles (at 300/1200/2400 with a slight serial-cable hardware modification which is thoroughly explained in the documentation). All DOS 3.3 compatible disk drives are supported--you can configure Apple-Net for one floppy drive, 6 floppy drives, or even a hard drive. (A ProDOS version of Apple-Net will be available soon--if you wish to be notified upon its release, leave [F]eedback to the Sysop with your name & address to let him know).

The nicest thing about Apple-Net, as many Sysops around the country keep telling me, is its ease of setup & use. Apple-Net was designed with the Sysop in mind. Everything is menu-driven and easy to use. For example, to add seperate boards for different topics, the Sysop just selects the Bulletin Board Editor, and enters the name of each board, the access level, and the drive where each board is located. The BBS then creates a board menu automatically and takes care of all the housekeeping. The same goes for General Files & Downloads.

Features of Apple-Net include:

```
   o  Extensive Electronic Mail section   o  Infinite number of sub-boards
   o  Downloading of programs             o  Uploading of files
   o  Complete message editor             o  User password protection
   o  Sub-menu'ed Feature Articles        o  Fast "Word Wrap" in Editor
   o  Old message auto-deletion           o  "Slash" message editor commands
   o  "Trick backspacing" in msgs.        o  Configurable "Spinning Cursor"
   o  Full upper/lowercase ability        o  Easy-to-use Sysop utilities
   o  Reliable carrier-loss detection     o  Easy and quick to set-up
```

Apple-Net BBS is guaranteed to never be "crashed" by anyone. The program is very sound, and any attempted "system crashers" are simply hung-up on by the system. Apple-Net, unlike other BBS programs such as GBBS, is also guaranteed to contain no "back-door". This was done for obvious reasons.

Apple-Net comes on a double sided disk with the programs on the front, and complete, comprehensive documentation on the back side which can be viewed on your computer screen or dumped out to your printer.

Updates may be made to Apple-Net periodically (minor modifications, bug fixes, new features, etc.) and will be announced on The Safehouse BBS. You may receive any updates tha are made to the program, free, by sending your Apple-Net disk and a dollar (for postage/handling).

```
I know its difficult to purchase software by just reading about it.  If you
would like to see Apple-Net in action, you may want to try one of these
bulletin board systems currently running Apple-Net:

    Stronghold North...612/588-7865    MCMLXXXV...........612/729-1985
    Chemist's Lab......602/577-1157    Black Fortress.....516/549-0268
    GolfSoft BBS.......612/941-8519    Gnome's Gnoll......206/334-0223
    Entertainment......507/282-8993    The Grotto.........218/727-2184

(Note that some of these BBS's may have made some minor modifications
 to the actual Apple-Net software which makes them a bit different.)

If you would like to purchase a copy of Apple-Net, just mail $55 (check,
cash, or money order) to:

   Dataware Corp.
   P.O. Box 17104
   Mpls, MN 55417

Your order will be shipped the day after it is received.

Please indicate the type of computer and modem you will be using for your BBS,
and the phone number that your BBS will be available at (if possible).

[<1-7>, ?=Menu, Q=Quit] :
```

```
================================================================================
DOCUMENT apples.txt
================================================================================
```

While trashing at Apple Computer Inc, among other things, we found a computer marked BROKEN with the following letter attached to it which described certain problems a user had found.

```
------------------------<: Letter reads :>-----------------------
```

Dear Sirs,

Enclosed is one defective Apple //e Computer which I returned because of the following defects:

1. The apple on the left side of the spacebar was not filled in. (looks like the person forgot to paint it.)
2. The TAB key never seems to produce the desired soft-drink. (I specifically ordered the COKE key!)
3. The ESC key never once helped me escape in any game I have played. (in fact, when playing Captain Goodnight, it caused my computer to hang, also I didn't know what ESC stood for until I found it in my manual.)
4. I assumed the CONTROL key, when pressed would switch control between joystick and keyboard, but no such luck.
5. The DELETE key refuses to delete the desired file on my disk. (The manual neglected to mention this feature. It seems as though a feature like this deserves more attention than is given to it by your company.)
6. Three of the keys on my computer have annoying pimples. (I tried using Clearasil to remove them, but to no avail, I was forced to use sandpaper.)
7. The four arrow keys refuse to print the desired arrow. I have tried repeatedly using the CONTROL & SHIFT key along

with the arsow key, but get no results except a strange movement of the cursor position.

8. The RESET key, located in an awkward position doesn't seem to have any effect on my computer when pressed.
   (I also noticed that this key has been reduced in size compared to the //+ series, no doubt to save cost.)

9. The Apple logo, which appears most frequently is technically incorrect. I think the painter had his paints mixed when adding this logo. Personally, I have never seen a multi-colored apple.

10. I think I have discovered a new key on the lower-left side of the keyboard. This mystery key serves no known function whatsoever and is not mentioned in the manual. I assumed this was a miniature monitor, because of the green light it emitted, however, I was unable to read anything displayed on this screen.
    (with the aid of a friend, I found a use for this key, by removing the plastic cover, it becomes a lamp. In your next revision, please increase the wattage to this lamp.)

11. Also, I found some problems with my apple monitor. The dials are missing and there is no volume control. Please send me a working monitor.

12. I also noticed that you shipped me two disk drives labelled 'DISK //'. I am thereby, returning one of the above mentioned disk drives for the proper 'DISK 1'.

13. I feel my keyboard has been damaged before shipment. The keys have a noticable dent in the middle.
    (Looks to me like an angry employee.)

14. I also noticed a key is missing! The 'Z' key cannot

be found anywhere on the computer at all.

15. Please disregard the above mentioned problem, I have managed to find the assumed lost 'Z' key.

16. Your ignorance in designing the back of my computer caused much damage to many of my peripheral cards. The slots behind the computer were much too small, and it took much strength to force my peripheral cards through them. (in fact, when installing my printer card, it took the help of my 3 sons and a wooden mallet.)

17. Also, I don't know where your engineers went to school, but where I am from, the alphabet does not resemble your key-board in any way.

18. I also suspect a problem with your key-manufacturing machine. The keys are not identical in size, in fact, there is a huge key, as you will notice, which has no label at all, and I have no idea of its use. Also the lettering on the keys is not centered. On my computer the lettering appears on the upper-left side, not in the center, like it should be.

19. Only because of my extensive educational background (graduated from the 8th grade and 1 month typing class) was I able to figure out the use of the SHIFT key. I suspected it had something to do with the different types of letters which appeared when I held down the shift key.

20. In regards to the above problem, I noticed the CAPS LOCK key (which tends to stick) will cause the SHIFT key to malfunction. It works with all other keys, except the letters!

21. Where did you people learn to count? Since when does '0' come after '9' ?! You seem to be confused between 0 and 10. Unless

you forgot the '1' before the '0'.

22. In regards to counting, your company boasts about there new computer having 64K, however, I was only able to find a single letter 'K' on the keyboard. (If this isn't false advertising, then I don't know what is!)
    I later found out through many hours of pondering on this question, that the 'K' stood for KEYS. (there are 64 of them on this computer!)
23. I found your manuals to be very inappropriate, besides being overly technical and hard to understand, the cover art clashed with the decore of my computer room.
24. After inserting my extended 80-column card, I was unable to extend the text page beyond 80 columns, therefore, the 80 column card is also inoperable.
25. I am upset about your company's morbid sense of humor and bad taste. I am referring to the hidden message which appears when the RESET key is hit along with the solid apple and the CONTROL key, 'KERNAL OK'. I find no humor in this comment about Colonel Sanders, everyone knows he has been dead for several years now.

As you can see, I have found many problems with your computer, and am very upset with your company. I demand a working computer and compensation for my troubles. I feel your customer support is insufficient and was very rude to me when I called and explained in detail the above mentioned problems. Your technical support person seemed to have problems with his telephone, I kept getting cut-off. After calling back 6 times I decided to send this letter.   Please give it your utmost attention and help to remedy this situation at once.

Sincerely,

Jethro McThorn

Okinfart, Nebraska

-----------------------<: End of letter :>---------------------

After reading this letter, we decided to investigate these matters. We called an Apple representative and they refused to comment on any questions regarding this letter. After finally giving up with Apple Computer Inc, we decided to turn to the media. We then gave 60 Minutes a ring, and informed them of our findings, they seemed very interested, and we mailed them a photo-copy of the letter. They are currently investigating this scandal and we should expect to see it on the air in a few weeks. We are working as consultants to 60 minutes and will inform you `bout any further findings. In the meantime, maybe you should reconsider purchasing another Apple Computer due to the defects and design flaws mentioned above.

B/R

```
==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==- ==-==
The Draco Tavern...........10 megs..................707/745-5805
Capital Connection.........10 megs (Soon!)...........916/448-3402
The Realm of Chaos.........Proving Grounds...........415-797-0121
==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==- ==-==
==============================================================================
DOCUMENT appleser.app
==============================================================================

       =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
       -                                  -
       =     //c Serial Ports -- J.A.K. //x      =
       -                                  -
       =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

-----------------------------------------------------------------
HAND CONTROL
-----------------------------------------------------------------
Connector   Signal
Pin Number  Name        Description

1           GAMESW1     Switch input (paddle #1)
2           +5          +5 VDC (Do not exceed 100 MA)
3           GND         System Ground
4,9         -           Not used for hand controllers
5,8         PDL0 & PDL1 Hand control inputs.  Each of
                        these must be connected to a 1K
                        pot connected to +5
6           N.C.        Not connected
7           GAMESW0     Switch input 0 (paddle #0)

-----------------------------------------------------------------
MOUSE CONNECTOR
-----------------------------------------------------------------
Connector   Signal
Pin number  Name        Description

1           MOUSEID     Mouse Identifier
2           +5V         +5VDC (Do not exceed 100 ma.)
3           GND         System Ground
4           X1          Mouse X-direction Indicator
5           X0          Mouse X-movement interrupt
6                       Mouse button
7           MSW         Mouse button
8           Y1          Mouse Y-direction indicator
9           Y0          Mouse Y-movement interrupt

-----------------------------------------------------------------
EXTERNAL POWER
-----------------------------------------------------------------
Connector   Signal
Pin Number  Name        Description

1,7                     Not Connected
2,3         Ground            Common electrical ground
4           Chassis     Chassis ground
5,6         +15V        +15VDC input to converter

=================================================================
        Description and Setup of Apple //C Serial Ports

The Apple //c serial ports are 5-pin DIN connectors.  Both Port 1 (Printer) and
```

Port 2 (Modem) have the same pin-out and signal description. Here are the cable descriptions for connecting them to the ImageWriter and the Apple Modem.

Apple //c Serial Port	ImageWriter	Apple Modem
DTR (1)	6 - DSR	6 - DTR
TXD (2)	3 - RCD	9 - TXD
GND (3)	7 - GND	3 - GND
RCD (4)	2 - TXD	5 - RCD
DSR (5)	20 - DTR	2 - DSR

```
                                           __
                  _ ______                |  ]
                   |_|   \                |  ]
               (5)         (1)        \ ________________ /   ]
                           )________________)    ]
               (4)         (2)        /              \       ]
                   (3)______ /                  |  ]
                                          |__]

             DIN type connector         DB type connector
```

Setting up the printer port from within a program on the //c is essentially the same as changing the settings on previous interface cards: after first directing output to the serial port (using PR#1 and PR#2 for Ports 1 and 2, respectively), the commands can then be sent to the serial port. Each command for Port 1 must be preceded by <CTRL-I>. The commands for Port 1 are:

Command	Description
nnn	Set line width (from 001 through 255): This command must be followed by an 'N' or a <CR>.
nnB	Set baud rate to value corresponding to nn.

nn	Rate	nn	Rate
01	50	09	1800
02	75	10	2400
03	110	11	3600
04	135	12	4800
05	150	13	7200
06	300	14	9600
07	600	15	19200
08	1200		

Command	Description
nB	Set Data Format to values corresponding to n.

n	Format
0	8 data 1 stop
1	7 data 1 stop
2	6 data 1 stop
3	5 data 1 stop
4	8 data 2 stop
5	7 data 2 stop
6	6 data 2 stop
7	5 data 2 stop

I	Echo output to screen.
K	Disable <LF> after <CR>.
L	Generate <LF> after <CR>.
nP	Set Parity corresponding to n.

n -	Parity
0	none
1	odd
2	none
3	even
4	none
5	mark (1)
6	none
7	space (0)

R	Reset Port 1 and exit from serial port 1 firmware.
S	Send a 233 millisecond Break character
Z	Zap (ignore) further command characters (until Control-Reset or PR#1). Do not format output or insert carriage returns into output stream.

Port 2 uses the same commands, with the differences and additions listed below. Each command for Port 2 must be preceded by a <CTRL-A>.

nnn	same
nnB	same
nD	same
I	same
K	same
L	same
nP	same
Q	Quit Terminal Mode
R	same
S	same
T	Enter Terminal Mode. Use this command after IN#2 only. If you follow this command by PR#2, the //c will echo input to output. (NOTE: If the other device is also echoing input to output, entering the first character will cause an infinite loop. Use <CTRL-RESET> to get

Z	same
Control-T	When issued from a remote device, this command puts the //c in terminal mode if IN#2 is already in effect. The
Control-R	When issued from a remote device, this command undoes the terminal mode command. If IN#2 and PR#2 are in effect, the remote keyboard and display become the input and output devices of the local //c. The command is the same as <CTRL-A> "Q" typed locally.

===

Description Of the Apple //C Video Expansion Port

The back panel of the Apple //c has a DB-15 connector for sophisticated video interfaces external to the computer. See table below for description of signals.

In the table, the column labled Deriv indicates what clock signals the video signals are derived from. LDPS, CREF and PRAS have a maximum delay of 30ns from the appropriate 14MHz rising edge. SEROUT is clocked out of a 74LS166 by the rising edge of 14M and has a maximum delay of 35ns. VIDD7 is driven from a 74LS374 and has a maximum delay of 28ns from the rising and (if 80 column) falling edges of phase1.

To align CREF so it is in the same phase at the beginning of every line, certain clock signals must be stretched. This stretch is for one 7M cycle (140ns), and occurs at the end of each video line. All timing signals except 14M, 7M and CREF are stretched.

WARNING!!! The signals at the DB-15 on the Apple //c are not the same as those on the Apple ///. Do not attempt to plug a cable intended for one into the other.

WARNING!!! Several of these signals, such as 14MHz, must be buffered within about four inches (10 cm) of the back panel connector - preferably inside a container directly connected to the back panel.

The Video Expansion Connector Pinouts

Pin	Deriv	Name	Description
1	phase1	TEXT	Video text signal from TMG; set to inverse of GR, except in double high-resolution mode
2		14M	14M master timing signal from the system oscillator
3	Q3	SYNC*	Display horizontal and vertical synchronization signal from IOU pin 39
4	PRAS	SEGB	Display vertical counter bit from IOU pin 4; in text mode indicates second low-order vertical counter; in graphics mode indicates low-resolution

```
        5         1VSOUND One-volt sound signal from pin 5 of
                        the audio hybrid circuit (AUD)

        6    14M    LDPS*   Video shift-register load enable from
                        pin 12 of TMG

        7    PRAS   WNDW*   Active display area blanking; includes
                        both horizontal and vertical blanking

        8         +12 V   Regulated +12 volts DC.; can drive
                        350mA

        9    14M    PRAS*   RAM row-address strobe from TMG pin 19

        10   PRAS   GR      Graphics mode enable from IOU pin 2

        11   14M    SEROUT* Serialized character-generator output
                        from pin 1 of the 74LS166 shift
                        register

        12        NTSC    Composit NTSC video signal from VID
                        hybrid chip

        13        GND     Ground reference and supply

        14   phase0 VIDD7   From 74LS374 video latch; causes
                        half-dot shift if high

        15   14M    CREF    Color reference signal from TMG pin 3;
                        3.58MHz
```

WARNING!!! Use caution--The maximum allowable current drain of +12V regulated power at the video expansion connector is 300 milliamps. If the external device draws more than this. it can damage the computer or cause the power supply to shut down.

===

Description Of Apple //C External Disk Port

The Apple //c external disk drive port is a DB-19 connector. The signals available at the port are as follows:

```
                     10 - WRPROT
         1 - GND
                     11 - SEEKPH0
         2 - GND
                     12 - SEEKPH1
         3 - GND
                     13 - SEEKPH2
         4 - GND
                     14 - SEEKPH3
         5 - +12V
                     15 - /WRREQ
         6 - + 5V
                     16 - NC
         7 - +12V
                     17 - /DR2
```

```
                8 - +12V
                            18 - RDDATA
                9 - /EXTINT
                            19 - WRDATA
```

CAUTION: This is not a recommendation by Apple to connect any but the Disk //c. Connecting any other disk drive will invalidate the Apple warranty.

==

Using the AppleSoft Sampler on a //c

Customers are finding that the Applesoft Sampler diskette (included with the Applesoft Tutorial), when used on an Apple //c, does not function as expected. Menus generated by this software are correct when used on an Apple //e, but there is a vertical displacement of one line between menu selection numbers and menu item descriptions when executed on an Apple //c.

The problem resides in two programs (CONVERTER and DISK.MENU) located on the Applesoft Sampler diskette. Load the "CONVERTER" program and list line number 625. This line currently reads;

```
     625 VTAB PEEK(37): IF COL80 THEN VTAB PEEK(1531)
```

Change this line to read:

```
     625 IF COL80 THEN VTAB PEEK(1531) : GOTO 630
```

and then add the following line:

```
     627 VTAB PEEK(37)
```

SAVE the file "CONVERTER".

Make the same changes to the file "DISK.MENU"

These changes will fix the problem.

We have been informed that these changes will soon be included in the final product. However, the problem exists in product that is already shipping. Our suggestion is to perform the above changes to the CONVERTER and DISK.MENU programs and give the fixed version of the program to those that need it.

NOTE: The following program will automatically update the Tutorial diskettes.

```
     100 D$ =  CHR$ (4): REM CTRL-D
     105  PRINT D$;"OPEN COMMAND.FILE"
     110  PRINT D$;"WRITE COMMAND.FILE"
     115 F$ = "DISK.MENU": GOSUB 200
     120 F$ = "CONVERTER": GOSUB 200
     125  PRINT "RUN DISK.MENU"
     130  PRINT D$;"CLOSE COMMAND.FILE"
     135  PRINT D$;"EXEC COMMAND.FILE"
     140  END
     200  PRINT "LOAD ";F$
     205  PRINT "625 IF COL80 THEN VTAB PEEK(1531):GOTO 630
     210  PRINT "627 VTAB PEEK(37)
     215  PRINT "UNLOCK ";F$
```

```
220  PRINT "SAVE ";F$
225  RETURN
```

```
================================================================================
DOCUMENT applesoft.tips
================================================================================

                       BEAGLE BROTHERS HINTS!
                       ----------------------

MAKE THE RESET KEY ACT LIKE CTRL-C (TRAPPABLE BY ONN ERR)
TYPE INTO YOUR PROGRAM: POKE 40286,35:POKE 40287,216
                        ON ERR GOTO 1000 (OR ANY LINE #)

MAKE THE RESET KEY BOOT WHEN PRESSED
FOR X=1011 TO 1015: POKE X,0: NEXT

AT SIGN (@) INSTEAD OF CTRL-D
POKE 43689,192

SCREEN SAVES
HI-RES PAGE 1-BASVE XXX,A$2000,L$2000
HI-RES PAGE 2-BSAVE XXX,A$2000,L$2000
LO-RES PAGE-BSAVE XXX,A$400,L$400
TEXT PAGE-BSAVE XXX,A$400,L$3FF

PREVENT CATALOG
POKE-21503,16

POKE INSTEAD OF FP COMMAND
POKE 2049,0:POKE 2050,0
PUT THESE AT THE END OF YOUR PROGRAM, IT'LL ERASE ITSELF

INVERSE, FLASH, & NORMAL WITH POKES
INVERSE-POKE 50,63  FLASH-POKE 50,127  NORMAL-POKE 50,255

LINE FINDS
WHAT LINE IS OPERATING, PRINT PEEK(117)+PEEK(118)*256
WHERE ON ERR WAS ENCOUNTERED- PRINT PEEK(118)+PEEK(119)*256

DOUBLE QUOTES IN A PRINT STATEMENT
TYPE:  10 Q$=CHR$(34)
       20 PRINT "THIS IS ";Q$"ILLEGAL.";Q$

CHANGE THE CATALOG TRACK-(CHEAP PROTECTION)
A) TYPE 'POKE 44033,XX' (WHERE XX IS NEW CATALOG TRACK)
B) INIT A NEW DISK
1) BOOT NORMAL DISK
2) LOAD A PROGRAM FROM THE DISK
3) TYPE 'POKE 44033,XX' (WHERE XX IS NEW CATALOG TRACK)
4) INSERT PROTECTED DISK (FROM STEP B)
5) SAVE THE PROGRAM
6) TYPE 'POKE 44033,17' (THE NORMAL #)
7) INSERT NORMAL DISK AND CONTINUE WITH STEP 2

DISABLE ON ERR FUNCTION
POKE 216,0

FREE MEMORY CHECK
PRINT FRE(0)+65536
```

```
TO MAKE SOMEONE BOOT DISK BEFORE RUNNING A PROGRAM
1) PUT IN BLANK DISK
2) TYPE 'POKE 47721,123'
3) TYPE 'NEW'
4) TYPE 'INIT HELLO'
5) NEAR BEGINNING OF PROGRAM PUT:
IF PEEK (47721) <> 123 THEN PRINT CHR$(4)"PR#1"; PEEK (43626)

           TIPS
        ----------
DIVIDE SECTOR COUNT BY 4 TO GET 'K' USED.

STOP WILL DO SAME AS END BUT WILL GIVE # OF LINE PROGRAM ENDED ON.

YOU CAN START A FILE WITH ANY CHARACTER WHO'S ASCII CHARACTER IS ABOVE 63

IF YOU'RE WRITING PROGRAMS THAT SOMEONE ELSE WILL SEE USE 'PRINT SPC(10)'
INSTEAD OF PRINT"          "

TO GET THE POSITIVE LOCATION OF AN ADDRESS ADD 65536 TO THE NUMBER.
EG.---> CALL -958 = CALL 64578

TO GET THE ALTERNATE CHARACTERS HOLD DOWN THE 'SHIFT', 'U', AND 'I' KEYS
AND TYPE :  'Y' FOR UNDERLINE
            'H' FOR BACKSLASH
            'J' FOR LEFT BRACKET
```

```
==============================================================================
DOCUMENT appswitc.app
==============================================================================

/----------------------------------------------------------------------------\
         Apple //e Soft Switch, Status, and other I/O locations
\----------------------------------------------------------------------------/

MEMORY MANAGEMENT SOFT SWITCHES

$C000 W     80STOREOFF  Allow page2 to switch video page1 page2
$C001 W     80STOREON   Allow page2 to switch main & aux video memory
$C002 W     RAMRDOFF    Read enable main memory from $0200-$BFFF
$C003 W     RAMDRON     Read enable aux memory from $0200-$BFFF
$C004 W     RAMWRTOFF   Write enable main memory from $0200-$BFFF
$C005 W     RAMWRTON    Write enable aux memory from $0200-$BFFF
$C006 W     INTCXROMOFF Enable slot ROM from $C100-$CFFF
$C007 W     INTCXROMON  Enable main ROM from $C100-$CFFF
$C008 W     ALZTPOFF    Enable main memory from $0000-$01FF & avl BSR
$C009 W     ALTZPON     Enable aux memory from $0000-$01FF & avl BSR
$C00A W     SLOTC3ROMOFF      Enable main ROM from $C300-$C3FF
$C00B W     SLOTC3ROMON Enable slot ROM from $C300-$C3FF

VIDEO SOFT SWITCHES

$C00C W     80COLOFF    Turn off 80 column display
$C00D W     80COLON     Turn on 80 column display
$C00E W     ALTCHARSETOFF     Turn off alternate characters
$C00F W     ALTCHARSETON      Turn on alternate characters
$C050 R/W   TEXTOFF     Select graphics mode
$C051 R/W   TEXTON            Select text mode
$C052 R/W   MIXEDOFF    Use full screen for graphics
$C053 R/W   MIXEDON     Use graphics with 4 lines of text
$C054 R/W   PAGE2OFF    Select panel display (or main video memory)
$C055 R/W   PAGE2ON     Select page2 display (or aux video memory)
$C056 R/W   HIRESOFF    Select low resolution graphics
$C057 R/W   HIRESON     Select high resolution graphics

SOFT SWITCH STATUS FLAGS

$C010 R7    AKD         1=key pressed     0=keys free    (clears strobe)
$C011 R7    BSRBANK2    1=bank2 available    0=bank1 available
$C012 R7    BSRREADRAM  1=BSR active for read   0=$D000-$FFFF active
$C013 R7    RAMRD       0=main $0200-$BFFF active reads  1=aux active
$C014 R7    RAMWRT            0=main $0200-$BFFF active writes 1=aux writes
$C015 R7    INTCXROM    1=main $C100-$CFFF ROM active 0=slot active
$C016 R7    ALTZP       1=aux $0000-$1FF+auxBSR    0=main available
$C017 R7    SLOTC3ROM   1=slot $C3 ROM active   0=main $C3 ROM active
$C018 R7    80STORE     1=page2 switches main/aux   0=page2 video
$C019 R7    VERTBLANK   1=vertical retrace on   0=vertical retrace off
$C01A R7    TEXT        1=text mode is active   0=graphics mode active
$C01B R7    MIXED       1=mixed graphics & text    0=full screen
$C01C R7    PAGE2       1=video page2 selected or aux
$C01D R7    HIRES       1=high resolution graphics   0=low resolution
$C01E R7    ALTCHARSET  1=alt character set on    0=alt char set off
$C01F R7    80COL       1=80 col display on     0=80 col display off
\============================================================================/
```

```
================================================================================
DOCUMENT bin.ii
================================================================================
```

.X:05

.LB:58

Binary II

Apple II Binary File Format

developed by

Gary B. Little

Version History

November 24, 1986 : Initial release.

Background

Transferring Apple II files in binary form to commercial information services like CompuServe, Delphi, GEnie, and The Source is, to put it mildly, a frustrating exercise. (For convenience, I'll refer to such services, and any other non-Apple II systems, as "hosts.") Although most hosts are able to receive a file's *data* in binary form (using the Xmodem protocol, for example), they don't receive the file's all-important attribute bytes. All the common Apple II operating systems, notably ProDOS, store the attributes inside the disk directory, not inside the file itself.

The ProDOS attributes are the access code, file type code, auxiliary type code, storage type code, date of creation and last modification, time of creation and last modification, the file size, and the name of the file itself. (All these terms are defined in Apple's "ProDOS Technical Reference Manual" or in the book "Apple ProDOS: Advanced

Features for Programmers" by Gary Little.) It is usually not possible to use a ProDOS file's data without knowing what the file's attributes are (particularly the file type code, auxiliary type code, and size). This means ProDOS files uploaded in binary form to a host are useless to those who download them. The same is true for DOS 3.3 and Pascal files.

Most Apple II communications programs use special protocols for transferring file attributes during a binary file transfer, but none of these protocols have been implemented by hosts. These programs are only useful for exchanging files with another Apple II running the same program.

At present, the only acceptable way to transfer an Apple II file to a host is to convert it into lines of text and send it as a textfile. Such a textfile would contain a listing of an Applesoft program, or a series of Apple II system monitor "enter" commands (e.g., 0300:A4 32 etc.). Someone downloading such a file can convert it to binary form using the Applesoft EXEC command.

The main disadvantage of this technique is that the text version of the file is over three times the size of the original binary file, making it expensive (in terms of time and $$$) to upload and download. It is also awkward, and sometimes impossible, to perform the binary-to-text or text-to-binary conversion.

The solution to the problem is to upload an encoded binary file which contains not just the file's data, but the file's attributes as well. Someone downloading such a file, say using Xmodem, can then use a conversion program to strip the attributes from the file and create a file with the required attributes.

To make this technique truly useful, however, the Apple II community must agree on a format for this encoded binary file. A variety of incompatible formats, all achieving the same general result, cannot be allowed to appear.

It is proposed that the Binary II format described in this document be adopted. What follows is a description of the Binary II format in sufficient detail to allow software developers to implement it in Apple II communications programs.

The Binary II File Format

The Binary II form of a standard file consists of a 128-byte file information header followed by the file's data. The data portion of the file is padded with nulls ($00 bytes), if necessary, to ensure the data length is an even multiple of 128. As a result, the Binary II form of a file is never more than 255 bytes longer than the original file.

The file information header contains four ID bytes, the attributes of the file (in ProDOS 8 form), and some control information. Here is the structure of the header:

Offset	Length	Contents
+0	1	ID byte: always $0A
+1	1	ID byte: always $47
+2	1	ID byte: always $4C
+3	1	access code
+4	1	file type code
+5	2	auxiliary type code
+7	1	storage type code
+8	2	size of file in 512-byte blocks
+10	2	date of modification
+12	2	time of modification

+14	2	date of creation
+16	2	time of creation
+18	1	ID byte: always $02
+19	1	[reserved]
+20	3	end-of-file (EOF) position
+23	1	length of filename/partial pathname
+24	64	ASCII filename or partial pathname
+88	23	[reserved, must be zero]
+111	1	ProDOS 16 access code (high)
+112	1	ProDOS 16 file type code (high)
+113	1	ProDOS 16 storage type code (high)
+114	2	ProDOS 16 size of file in blocks (high)
+116	1	ProDOS 16 end-of-file position (high)
+117	4	disk space needed
+121	1	operating system type
+122	2	native file type code
+124	1	phantom file flag
+125	1	data flags
+126	1	Binary II version number
+127	1	number of files to follow

Multi-byte numeric quantities are stored with their low-order bytes first, the same order expected by ProDOS. All reserved bytes must be set to zero; they may be used in future versions of the protocol.

To determine the values of the attributes to be put into a file information header for a ProDOS file, you can use the ProDOS GET_FILE_INFO and GET_EOF MLI commands.

Note: Some file attributes returned by ProDOS 16 commands are one or two bytes longer than the attributes returned by the corresponding ProDOS 8 commands. At present, these extra bytes are always zero, and probably will remain zero forever. In any event, place the extra bytes returned by ProDOS 16 in the header at +114 to +119. ProDOS 8 communications programs should zero these header locations.

The "disk space needed" bytes contain the number of 512-byte disk blocks the files inside the Binary II file will occupy after they've been removed from the Binary II file. (The format of a Binary II file containing multiple files is described below.) If the number is zero, the person uploading the file did not bother to calculate the space needed. The "disk space needed" must be placed in the file information header for the first file inside the Binary II file; it can be set to zero in subsequent headers. A downloading program can inspect "disk space needed" and abort the transfer immediately if there isn't enough disk free space.

The value of the "operating system type" byte indicates the native operating system of the file:

```
        $00 = ProDOS 8, ProDOS 16, or SOS
        $01 = DOS 3.3
        $02 = Pascal
        $03 = CP/M
        $04 = MS-DOS
```

Note that even if a file is not a ProDOS file, the attributes in the file information header, including the name, must be inserted in ProDOS form. Instructions on how to do this for DOS 3.3 files are given later in this document. Similar considerations apply for the files of other operating systems.

The "native file type code" has meaning only if the "operating system type" is non-zero. It is set to the actual file type code assigned to the file by its native operating system. (Some operating systems, such as CP/M and MS-DOS, do not use file type codes, however.) Contrast this with the file type code at +4, which is the closest equivalent ProDOS file type code. The "native file type code" is needed to distinguish files which have the same *ProDOS* file type, but which may have different file types in their native operating system. Note that if the file type code is only byte long (the usual case), the high-order byte of "native file type code" is set to zero.

The "phantom file flag" byte indicates whether a receiver of the Binary II file should save the file which follows (flag is zero) or ignore it (flag is non-zero). It is anticipated that some communications programs will use phantom files to pass non-essential explanatory notes or encoded information which would be understood only by a receiver using the same communications program. Such programs must not rely on receiving a phantom file, however, since this would mean they couldn't handle Binary II files created by other communications programs.

The first two bytes in a phantom file *must* contain an ID code unique to the communications program. Developers must obtain ID codes from Gary Little to ensure uniqueness (see below for his address). Here is a current list of approved ID codes for phantom files used by Apple II communications programs:

```
        $00 $00  =  [generic]
        $00 $01  =  Point-to-Point
        $00 $02  =  Tele-Master Communications System
```

Developers of communications programs are responsible for defining and publishing the structures of their phantom files.

The ID bytes appear in the first two bytes of the phantom file.

Phantom files having a generic ID code of zero must contain lines of text terminated by a $00 byte. The text must begin at the third byte in the file.

The "data flags" byte is a bit vector indicating whether the data portion of the Binary II file has been compressed, encrypted, or packed. If bit 7 (the high-order bit) is set to 1, the file is compressed. If bit 6 is 1, the file is encrypted. If bit 0 is 1, the file is a sparse file that is packed. A Binary II downloading program can examine this byte and warn the user, when necessary, that the file must be expanded, decrypted, or unpacked. The person uploading a Binary II file may use any convenient method for compressing, encrypting, or packing the file but is responsible for providing instructions on how to restore the file to its original state.

This initial release of Binary II has a "Binary II version number" of $00.

Handling Multiple Files

An appealing feature of Binary II is that a single Binary II file can hold multiple disk files, making it easy to keep a group of related files "glued" together when they're sent to a host.

The structure of a Binary II file containing multiple disk files is what you might expect: it is a series of images of individual Binary II files. For example, here is the general structure of a Binary II file containing three disk files:

```
 start                                                           end
 -----------------------------------------------------------------
 | Header #1 | #1 Data | Header #2 | #2 Data | Header #3 | #3 Data |
 -----------------------------------------------------------------
   +127 = 2              +127 = 1              +127 = 0
```

The data areas following each header end on a 128-byte boundary.

The "number of files to follow" byte (at offset 127) in the file information header for each disk file contains the number of disk files that follow it in the Binary II file. It will be zero in the header for the last disk file in the group.

Filenames and Partial Pathnames

Notice that you can put a standard ProDOS filename or a partial pathname in the file information header (but never a complete pathname). *Beware!* Don't use a partial pathname unless you've included, earlier on in the Binary II file, file information headers for each of the directories referred to in the partial pathname. Such a header must have its "end of file position" bytes set to zero, and no data blocks for the subdirectory file must follow it.

For example, if you want to send a file whose partial pathname is HELP/GS/READ.ME, first send a file information header defining the HELP/ subdirectory, then one defining the HELP/GS/ subdirectory. If you don't, someone downloading the Binary II file won't be able to convert it because the necessary subdirectories will not exist.

Filename Convention

Whenever a file is sent to a host, the host asks the sender to provide a name for it. If it's a Binary II file, the name provided should end in .BNY so that its special form will be apparent to anyone viewing a list of filenames.

Identifying Binary II Files

You can determine if a file is in Binary II form by examining the ID bytes at offsets +0, +1, +2, and +18 from the beginning of the file. They must be $0A, $47, $4C, and $02, respectively.

Once you identify a Binary II file, you can use the data in the file information header to create and open a ProDOS file with the correct name and attributes (using the CREATE, OPEN and SET_FILE_INFO commands), transfer the file data in the Binary II file to the ProDOS file, set the ProDOS file size (with SET_EOF), then close the ProDOS file. You would repeat this for each file contained inside the Binary II file.

Note: The number of 128-byte data blocks following the file information header must be derived from the "end-of-file position" attribute (EOF) not the "size of file in blocks" attribute. Calculate the number by dividing EOF by 128 and adding one to the result if EOF is not 0 or an exact multiple of 128.

Exception: If the file information header defines a subdirectory (the file type code is 15), simply CREATE the subdirectory file. Do not OPEN it and do not set its size with SET_EOF.

Ideally, all this conversion work will be done automatically by a communications program during an Xmodem (or other binary protocol) download. If not, a separate conversion program will have to be run after the Binary II file has been received and saved to disk. Gary Little has published a public domain program, called BINARY.DWN, that will do this for you. (A related program, BINARY.UP, combines multiple ProDOS files into one Binary II file which can then be uploaded to a host.)

DOS 3.3 Considerations

With a little extra effort, you can also convert DOS 3.3 files to Binary II form. This involves translating the DOS 3.3 file attributes to the corresponding ProDOS attributes so that you can build a proper file information header. Here is how to do this:

(1) Set the name to one that adheres to the stricter ProDOS naming rules.

(2) Set the ProDOS file type code, auxiliary type code, and access code to values which correspond to the DOS 3.3 file type:

DOS 3.3 file type	ProDOS file type	ProDOS aux type	ProDOS access
$00 (T)	$04 (TXT)	$0000	$E3
$80 (*T)	$04 (TXT)	$0000	$21
$01 (I)	$FA (INT)	$0C00	$E3
$81 (*I)	$FA (INT)	$0C00	$21
$02 (A)	$FC (BAS)	$0801	$E3
$82 (*A)	$FC (BAS)	$0801	$21
$04 (B)	$06 (BIN)	(*)	$E3
$84 (*B)	$06 (BIN)	(*)	$21
$08 (S)	$06 (BIN)	$0000	$E3
$88 (*S)	$06 (BIN)	$0000	$21
$10 (R)	$FE (REL)	$0000	$E3
$90 (*R)	$FE (REL)	$0000	$21
$20 (A)	$06 (BIN)	$0000	$E3
$A0 (*A)	$06 (BIN)	$0000	$21
$40 (B)	$06 (BIN)	$0000	$E3
$C0 (*B)	$06 (BIN)	$0000	$21

(*) Set the aux type for a B file to the value stored in the first two bytes of the file (this is the default load address).

(3) Set the storage type code to $01.

(4) Set the size of file in blocks, date of creation, date of modification, time of creation, and time of modification to $0000.

(5) Set the end-of-file position to the length of the DOS 3.3 file, in bytes. For a B file (code $04 or $84), this number is stored in the third and fourth bytes of the file. For an I file (code $01 or $81) or an A file (code $02 or $82), this number is stored in the first and second bytes of the file.

(6) Set the operating system type to $01.

(7) Set the native file type code to the value of the DOS 3.3 file type code.

Attribute bytes inside a DOS 3.3 file (if any) must *not* be included in the data portion of the Binary II file. This includes the first four bytes of a B (Binary) file, and the first two bytes of an A (Applesoft) or I (Integer BASIC) file.

Acknowledgements

Thanks to Glen Bredon for suggesting that partial pathnames be allowed in file information headers. Thanks also to Shawn Quick for suggesting the "phantom file" byte, to Scott McMahan for suggesting the compression and encryption bits in the "data flags" byte, and to William Bond for suggesting the "disk space needed" bytes. Finally, a big thank you to Neil Shapiro, Chief Sysop of MAUG, for supporting the development of the Binary II format and helping it become a true standard.

Feedback and Support

Send any comments or questions concerning the Binary II file format to:

Gary B. Little

#210 - 131 Water Street

Vancouver, British Columbia

Canada V6B 4M3

(604) 681-3371

CompuServe : 70135,1007

Delphi   : GBL

MCI Mail   : 658L6

Gary developed the Point-to-Point telecommunications program published by Pinpoint Publishing. He has also written several books on how to program Apple computers: "Inside the Apple IIe," "Inside the Apple IIc," "Apple ProDOS: Advanced Features for Programmers," and "Mac Assembly Language: A Guide for Programmers." He is currently a Contributing Editor for A+ magazine and writes A+'s monthly Rescue Squad column. Gary has also published articles in Nibble, Micro, Call -A.P.P.L.E, and Softalk.

==
DOCUMENT bitsbaud.doc
==

BITS, BAUD RATE, AND BPS
Taking the Mystery Out of Modem Speeds
by Michael A. Banks

Modem transmission speed is the source of a lot of confusion, even among otherwise informed computer and modem users. The root of the problem is the fact that the terms "baud" and "bits per second" are used interchangeably and indiscriminately. I strongly suspect this is a result of the fact that it's easier to say "baud" than "bits per second," though misinformation has a hand in it, too.

If you've ever found yourself confused by the relationship between bits and baud rate, or if you think that a modem's baud rate is the same as the number of bits or characters it transmits per second, please read this article carefully; I guarantee to clear up the confusion and disabuse you of any false concepts ...

Bits per second (bps)

Bits per second is a measure of the number of data bits (digital 0's and 1's) transmitted each second in a communications channel. This is sometimes referred to as "bit rate."

Individual characters (letters, numbers, etc.), also referred to as bytes, are composed of several bits.

While a modem's bit rate is tied to its baud rate, the two are not the same, as explained below.

Baud rate

Baud rate is a measure of the number of times per second a signal in a communications channel varies, or makes a transition between states (states being frequencies, voltage levels, or phase angles). One baud is one such change. Thus, a 300-baud modem's signal changes state 300 times each second, while a 600-baud modem's signal changes state 600 times per second. This does not necessarily mean that a 300-baud and a 600-baud modem transmit 300 and 600 bits per second, as you'll learn in a few lines.

Determining bits per second

Depending on the modulation technique used, a modem can transmit one bit--or more or less than one bit--with each baud, or change in state. Or, to put it another way, one change of state can transmit one bit--or more or less than one bit.

As I mentioned earlier, the number of bits a modem transmits per second is directly related to the number of bauds that occur each second, but the numbers are not necessarily the same.

To illustrate this, first consider a modem with a baud rate of 300, using a transmission technique called FSK (Frequency Shift Keying, in which four different frequencies are turned on and off to represent digital 0 and 1 signals from both modems). When FSK is used, each baud (which is, a gain, a change in state) transmits one bit; only one change in state is required to send a bit. Thus, the modem's bps rate is also 300:

```
     300 bauds per second X 1 bit per baud  =  300 bps
```

Similarly, if a modem operating at 1200 baud were to use one change in state to send each bit, that modem's bps rate would be 1200. (There are no 1200 baud modems, by the way; remember that. This is only a demonstrative and hypothetical example.)

Now, consider a hypothetical 300-baud modem using a modulation technique that requires two changes in state to send one bit, which can also be viewed as 1/2 bit per baud. Such a modem's bps rate would be 150 bps:

```
       300 bauds per second X 1/2 baud per bit  =  150 bps
```

To look at it another way, bits per second can also be obtained by dividing the modem's baud rate by the number of changes in state, or bauds, required to send one bit:

```
        300 baud
     ---------------  =  150 bps
     2 bauds per bit
```

Now let's move away from the hypothetical and into reality, as it exists in the world of modulation.

First, lest you be misled into thinking that "any 1200 baud modem" should be able to operate at 2400 bps with a two-bits-per-baud modulation technique, remember that I said there are no 1200 baud modems. Medium- and high-speed modems use baud rates that are lower than their bps rates. Along with this, however, they use multiple-state modulation to send more than one bit per baud.

For example, 1200 bps modems that conform to the Bell 212A standard (which includes most 1200 bps modems used in the U.S.) operate at 300 baud and use a modulation technique called phase modulation that transmits four bits per baud. Such modems are capable of 1200 bps operation, but not 2400 bps because they are not 1200 baud modems; they use a baud rate of 300. So:

```
     300 baud X 4 bits per baud  =  1200 bps

                                 or

          300 baud
     -----------------  =  1200 bps
      1/4 baud per bit
```

Similarly, 2400 bps modems that conform to the CCITT V.22 recommendation (virtually all of them) actually use a baud rate of 600 when they operate at 2400 bps. However, they also use a modulation technique that transmits four bits per baud:

```
     600 baud X 4 bits per baud  =  2400 bps

                                or

          600 baud
     -----------------  = 2400 bps
      1/4 baud per bit
```

Thus, a 1200-bps modem is not a 1200-baud modem, nor is a 2400-bps modem a 2400-baud modem.

Now let's take a look at 9600-bps modems. Most of these operate at 2400 baud, but (again) use a modulation technique that yields four bits per baud. Thus:

```
     2400 baud X 4 bits per baud  =  9600 bps

                                or

          2400 baud
     -----------------  =  9600 bps
      1/4 baud per bit
```

Characters per second (cps)

Characters per second is the number of characters (letters, numbers, spaces, and symbols) transmitted over a communications channel in one second. Cps is often the bottom line in rating data transmission speed, and a more convenient way of thinking about data transfer than baud- or bit-rate.

Determining the number of characters transmitted per second is easy: simply divide the bps rate by the number of bits per character. You must of course take into account the fact that more than just the bits that make up the binary digit representing a character are transmitted when a character is sent from one system to another. In fact, up to 10 bits may be transmitted for each character during ASCII transfer, whether 7 or 8 data bits are used. This is because what are called start- and stop-bits are added to characters by a sending system to enable the receiving system to determine which groups of bits make up a character. In addition, a system usually adds a parity bit during 7-bit ASCII transmission. (The computer's serial port handles the addition of the extra bits, and all extra bits are stripped out at the receiving end.)

So, in asynchronous data communication, the number of bits per character is usually 10 (either 7 data bits, plus a parity bit, plus a start bit and a stop bit, or 8 data bits plus a start bit and a stop bit). Thus:

```
             300 bps
     ----------------------  =  30 characters per second
      10 bits per character

            1200 bps
     ----------------------  =  120 characters per second
      10 bits per character

            2400 bps
     ---------------------  =  240 characters per second
      10 bits per character
```

Common speeds

The most commonly-used communications rates for dial-up systems (BBSs and online services like CompuServe, DELPHI, and GEnie) are 300, 1200, and 2400 bps. A few older systems--especially Telex systems--communicate at 110 bps, but these are gradually going the way of the dinosaur. 4800 and 9600 bps modems are generally available, but few online services or BBSs accommodate them. This will be changing in the near future, however, with the cost of high-speed modem technology decreasing as the demand for it increases.

Modems with even higher bps rates are manufactured (19,200 and up) but these are not used with dial-up systems; the upper limit on asynchronous data transmission via voice-grade telephone lines appears to be 9600 bps. The use of higher transmission rates requires special dedicated lines that are "conditioned" (i.e., shielded from outside interference) as well as expensive modulation and transmission equipment.

#

If you found this article useful, you may want to pick up a copy of the book from which it was excerpted:

THE MODEM REFERENCE
by Michael A. Banks
Published by Brady Books/Simon & Schuster

In addition to explaining the technical aspects of modem operation, communications software, data links, and other elements of computer communications, the book provides detailed, illustrated "tours" of major online services such as UNISON, CompuServe, DELPHI, BIX, Dow Jones News/Retrieval, MCI Mail, the PRODIGY service, and others. It also contains information on using packet switching networks and BBSs, as well as dial-up numbers for various networks and BBSs.

You'll also find hands-on guides to buying, setting up, using, and troubleshooting computer communications hardware and software. (And the book "supports" all major microcomputer brands.) THE MODEM REFERENCE is available at your local B. Dalton's, WaldenSoftware, Waldenbooks, or other bookstore, either in stock or by order. Or, phone 800-624-0023 to order direct.

Want the lowdown on getting more out of your word processor? Read the only book on word processing written by writers, for writers: WORD PROCESSING SECRETS FOR WRITERS, by Michael A. Banks & Ansen Dibel (Writer's Digest Books). WORD PROCESSING S
ZOaG&:∞xG
 WRITERS is available at your local B. Dalton's, Waldenbooks, or other bookstore, either in stock or by order. Or, phone 800-543-4644 (800-551-0884 in Ohio) to order direct.

Do you use DeskMate 3? Are you getting the most out of the program? To find out, get a copy of GETTING THE MOST OUT OF DESKMATE 3, by Michael A. Banks. Published by Brady Books/Simon & Schuster, it is available in your local Tandy/Radio Shack, WaldenSoftware, or Waldenbooks store now. Or, phone 800-624-0023 to order direct.

```
                  Other books by Michael A. Banks
UNDERSTANDING FAX & E-MAIL (Howard W. Sams & Co.)ZA∑‹∑>& F9ch Books)]X
 [For more information, contact:]X
 [Michael A. Banks]X
 [P.O. Box 312]X
 [Milford, OH  45150]X
 aOE MAUSER: Movel; Baen Books)
T PRICES (w/Mack Reynolds; SF novel; Baen Books)
COUNTDOWN: THE COMPLETE GUIDE TO MODEL ROCKETRY (TAB Books)
THE ROCKET BOOK (w/Robert Cannon; Prentice Hall Press)
SECOND STAGE: ADVANCED MODEL ROCKETRY (Kalmbach Books)
     For more information, contact:
                        Michael A. Banks
                          P.O. Box 312
                       Milford, OH  45150
```

```
==============================================================================
DOCUMENT bootl-6
==============================================================================

---------------------------------------
--  How to modify the 16k Ram Board  --
             By: Axe Man
---------------------------------------

WRITE PROTECT:
    LIFT PIN #3 FROM U18 CHIP & CONNECT
    TO ONE SIDE OF SWITCH.
    CONNECT SOCKET AND PIN #13 74LS175
    TO CENTER OF SWITCH
    CONNECT TOP OF R3 TO OTHER SIDE OF
    THE SWITCH

R3---------------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #13----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------0
U18

CHANGES FOR RAM & ROM
    LIFT PIN #3 FROM U14 CHIP & CONNECT
    TO ONE SIDE OF SWITCH
    CONNECT SOCKET AND PIN #5 74LS175
    TO CENTER OF SWITCH
    CONNECT GROUND TO OTHER SIDE

GROUND-----------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #5-----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------O
U14

* * * * * * * W A R N I N G * * * * * *
THIS IS DONE AT YOUR OWN RISK
IT WILL VOID YOUR GUARANTEE
WE ASSUME NO RESPONSIBILITY FOR RESULTS
* * * * * * * W A R N I N G * * * * * *

IT SEEMS THERE'S A DEMAND FOR A W/P
SWITCH ON THE ANDROMEDA -- SO HERE IT
IS ...

 LOCATED ON THE ANDROMEDA RAM CARD IS
 A PIN NUMBER 25 WHICH HAPPENS TO BE
 THE POWER (+5V) PIN. IF THIS PIN IS
 FOLLOWED ONTO THE PC BOARD, THERE WILL
 BE TWO RESISTORS (SMALL TUBE-LIKE
THINGS WITH COLOR BANDS AND ONE LEAD
OUT OF EACH END). AT ONE END THE POWER
WILL GO INTO THIS RESISTOR, AT THE OTHE
R ANOTHER TRACE WILL GO OFF TO SOME
OF THE OTHER ELECTRONICS ON THE BOARD.
WE WANT TO USE THE END THAT HAS THE
TRACES GOING TO OTHER CHIPS ON THE
BOARD. (CALL THIS POINT #1 (USE EITHER
RESISTOR - THERE ARE TWO)). POINT NUMBE
R TWO IS WHERE PIN 18 FROM THE APPLE
CONNECTOR (7 PINS DOWN FROM 25 ON THE
SAME SIDE) ENTERS ONTO THE PC BOARD
AND IMMEDIATELY GOES THROUGH TO THE
OTHER SIDE (AFTER ABT 1/2 "). THIS
IS POINT #2.  IF YOU TRACE WHERE TH
E THING COMES OUT ON THE OTHER SIDE,
YOU'LL FIND OUT THAT IT POPS BACK ON
THE SIDE IT STARTED FROM ABOUT 1/2"
LATER... THIS LITTLE LINK IS WHEERE WE
CUT THE TRACE TO INSERT THE SWITCH.
OK, WE CUT THE TRACE BETWEEN THE TWO
POINTS THAT IT GOES THROUGH THE PC
BOARD. LABEL THE OTHER PLACE WHERE THE
TRACE GOES THROUGH POINT#3.  NOW WE
WILL ATTACH AN SPDT SWITCH TO THE BOARD
 SOLDER ONE WIRE TO POINT 3, AND ATTACH
IT TO THE CENTER TERMINAL OF THE SWITCH
THEN SOLDER A WIRE TO POINT 1 AND
ATTACH IT TO EITHER SIDE OF THE CENTER
SWITCH. LASTLY, TAKE A WIRE AND SOLDER
IT TO POINT 2 AND THEN TO THE UNUSED
PIN ON THE SWITCH. THERE YOU HAVE IT!
WHEN THE SWITCH HANDLE IS ON THE SAME
SIDE AS THE WIRE FROM POINT #1, REG-
ULAR OPERATION WILL TAKE PLACE. IF THE
SWITCH IS THROWN IN THE OTHER DIRECTION
THE CARD WILL BE WRITE PROTECTED.
(*PLEASE NOTE THAT THIS MODIFICATION
 WILL VOID YOUR WARRANTY AND THAT THE
USER ASSUMES AND WILL BE RESPONSIBLE
FOR ALL RISKS AND DAMAGES INCURRED IN
THE MAKING OR THE USE OF THIS MOD-
IFICATION, AND THAT THIS MODIFICATION
IS NOT GUARANTEED TO BE SUITABLE FOR
ANY PARTICULAR PURPOSE*)
----------------------------------------

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

FOR YE WITH SCANNERS IN NEW JERSEY-
N.J. BELL SECURITY CAN BE FOUND AT

462.55   462.575   462.600   462.625
```

```
462.65   462.675   462.700   462.725

ABOVE FREQUENCYS ARE IN MHZ.

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THATS RIGHT! IF YA WANT A CATALOG OF YE
BELL DOCUMENTATION CATALOG FER FREE-
THEN CALL 1-800-432-6600

THIS CATALOG LISTS AT&T DOCS FOR SALE!

IF YA GET ANY-LETS TRADE-CALL ME VIA
VOICE AT 503-592-4461

NUFF SAID-
BOOTLEG

OK-TO ALL OF YOU FOOLISH ENOUGH TO
ASK

     NO-NO-NO FREE SAMPLE ISSUES!

BUT BACK ISSUES ARE AVAILABLE FOR
$10 EACH.

IF YOU HAVE PROBLEMS WITH ANY DISK
NOT WORKING-MAIL IT BACK WITH AN
EXPLANATION OF THE PROBLEM.
ILL REMAIL A FRESH COPY UPON GETTING
THE OLD ONE.

AND TELL YOUR FRIENDS-NO FREE SAMPLES
(UNLESS THEY FEEL LIKE MAILING FREE
SAMPLES OF 10 DOLLAR BILLS FIRST!)

NUFF SAID-
BOOTLEG

MSG LEFT BY: SALLY RIDE
DATE POSTED:

AN INTERESTING MESSAGE POSTED ON TEXCON
CAUGHT MY EYE..IN LIGHT OF THE NAZI BBS
THE MOLESTERS BBS AND THE BUST IN N.J..
A NEW FEDERAL LAW THAT WOULD OUTLAW OR
SEVERELY RESTRICT BBS ACTIVITY IS BEING
PUSHED THROUGH CONGRESS, ACCORDING TO
TWO SOURCES THE METAL DETECTOR AND JOHN
EDENS. ANOMG THE RESTRICTIONS DISCUSSED
ARE: *REGISTRATION OF ALL BBS AS PUBLIC
      UTILITIES.
     *SYSOPS REQUIRED TO KEEP LOGS OF
      ALL USERS "VERIFIED" NAMES AND
      ADDRESSES.
     *SYSOPS REQUIRED TO KEEP LOGS OF
      ALL MESSAGE POSTINGS AND TIMES OF
      POSITNG
     *CRIMINAL PENALTIES FOR SYSOPS WHO
      ALLOW ILLEGAL MESSAGE POSTINGS
      WHETHER OR NOT THEY HAVE KNOW OF
      THE CONTENT OR HAVE HAD A CHANCE
      TO REMOVE IT.
     *BBS USERS WOULD BE "REQUIRED" TO
      USE THEIR LEGAL NAMES.
I HAVE NO WAY TO BE SURE THIS DATA IS
CORRECT, BUT I SUSPECT IT IS CLOSE TO
THE TRUTH. CONTACT YOUR CONGRESSIONAL
REPRESENTATIVE, THEY ALMOST ALL HAVE
800 #'S, AND FIND OUT AND EXPRESS YOUR
OPINION, TOO, WHILE YOU HAVE THE CHANCE
SALLY RIDE:::SPACE CADET

MSG LEFT BY: THE WARLOCK LORD
DATE POSTED:

IS SO DAMN UNCONSTITUTIONAL IT MAKES ME SICK.  THE SYSOP SHOULD NOT BE HELD
RESPONSIBLE FOR MESSAGES, HE IS PROTECTED BY THE FIRST AMENDMENT.  IT'LL
NEVER PASS.

THE WARLOCK LORD
WHEW-

WHAT A YEAR SO FAR-
BOARDS BEING BUSTED THROUGHOUT THE U.S.
NEW COMPUTER CRIME LAWS BEING ABUSED BY
COPS IN MOST STATES (THOUGH MOST OF THESE LAWS CERTAINLY WILL PROVE
UN-CONSTITUTIONAL IF EVER TAKEN TO THE SUPREME COURT).
AND-NEW FEDERAL LAWS IN THE FEDERAL SUB-COMMITTEES WAITIN TO BE VOTED ON!
ALSO,YE SECRET SERVICE HAS TAKEN OVER
FEDERAL COMPUTER INVESTIGATIONS FROM
THE FBI.

SEEMS LIKE EVERY PREDICTION I MADE IN
EARLIER ISSUES HAS COME TRUE!

NOW LET ME TELL YA WHATS HAPPENIN AND
WHY. THE FEDS KNOW EVERYTHING IS GOIN
COMPUTER IN THE NEAR FUTURE,AND NATURALLY LAW ENFORCEMENT AGENCYS WANTED THE
WORST LAWS POSSIBLE ON THE BOOKS SO
THEY COULD RUN AMOK WITHOUT ANY CONCERN
OF OUR CONSTITUTIONAL RIGHTS.
SINCE OUR LEGISLATORS KNOW LITTLE OF
COMPUTER HAPPENINGS,YE SNEAKY FEDS
DECIDED TO RAID VIRTUALLY EVERY KID
THAT RAN A GOOD BOARD THIS SPRING.
NATURALLY,THEY SPENT MEGA-THOUSANDS
ON THESE RAIDS AND THEN WERE KIND
ENOUGH TO MIS-INFORM THE PRESS AS
TO FABLES LIKE HACKERS MOVIN SATELITES,
ETC.NOW WHEN OUR INK HUNGRY MEDIA GOT
HOLD OF ALL THESE LIES, THEY TRUMP
```

```
EM UP EVEN FURTHER IN A BLITZ THAT
INCLUDED FRONT PAGE HEADLINES,UPI,
AND NATIONAL TV.
 O.K. SO HERE SITS MR & MRS CITIZEN
BELIEVIN THAT GARBAGE AN CALLIN MR
CONGRESSMAN SCREAMIN BLOODY MURDER.
 AN YA CAN GUESS WHATS HAPPENIN NOW-
YEP-THEYRE TRYIN TO DEPRIVE US OF
CONSTITUTIONAL RIGHTS (1ST & 4TH
AMMENDMENTS) THROUGH NEW PENDING
LEGISLATION......

NOW YA SAY-WHAT CAN BE DONE?

OK-WRITE EVERY CONGRESSMAN IN YOUR
STATE AND ALSO OTHER STATES AN EVEN
THE PRESIDENT TELLIN THEM YOU DONT
WANT ANY LAWS PASSED THAT WILL
INFRINGE ON YOUR RIGHTS RE COMPUTERS.
MASS PRODUCE THOSE LETTERS & SEND EM
OUT NOW-THESE BAD LAWS CAN ONLY BE
STOPPED BY DOIN SOMETHING RIGHT AWAY.

ALSO-BITCH ABOUT SPENDING ALL OUR
TAX MONEY HARASSING KIDS.THERE IS
A LOT OF CROOKS RUNNIN ROUND THEY
SHOULD BE AFTER WITH THOSE FEDERAL
AGENTS.

AND MAKE UP PETITIONS SIGNED BY
ANYONE FROM ADULTS TO YOUR CLASSMATES
TO SEND IN ALSO. HECK-SEND THOSE
LETTERS IN EVERY WEEK TILL WE GET
RESULTS.ALSO-POST THIS FILE EVERYWHERE.



         THE BOOTLEGGER MAGAZINE
         1080 HAYS CUT-OFF ROAD
         CAVE JCT.OR.97523

NUFF SAID-
BOOTLEG
---------------------------------------
--  How to modify the 16k Ram Board  --
            By: Axe Man
---------------------------------------

WRITE PROTECT:
    LIFT PIN #3 FROM U18 CHIP & CONNECT
    TO ONE SIDE OF SWITCH.
    CONNECT SOCKET AND PIN #13 74LS175
    TO CENTER OF SWITCH
    CONNECT TOP OF R3 TO OTHER SIDE OF
    THE SWITCH

R3--------------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #13----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------0
U18

CHANGES FOR RAM & ROM
    LIFT PIN #3 FROM U14 CHIP & CONNECT
    TO ONE SIDE OF SWITCH
    CONNECT SOCKET AND PIN #5 74LS175
    TO CENTER OF SWITCH
    CONNECT GROUND TO OTHER SIDE

GROUND-----------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #5-----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------O
U14

* * * * * * * W A R N I N G * * * * * *
THIS IS DONE AT YOUR OWN RISK
IT WILL VOID YOUR GUARANTEE
WE ASSUME NO RESPONSIBILITY FOR RESULTS
* * * * * * * W A R N I N G * * * * * *

IT SEEMS THERE'S A DEMAND FOR A W/P
SWITCH ON THE ANDROMEDA -- SO HERE IT
IS ...

 LOCATED ON THE ANDROMEDA RAM CARD IS
 A PIN NUMBER 25 WHICH HAPPENS TO BE
 THE POWER (+5V) PIN. IF THIS PIN IS
 FOLLOWED ONTO THE PC BOARD, THERE WILL
 BE TWO RESISTORS (SMALL TUBE-LIKE
THINGS WITH COLOR BANDS AND ONE LEAD
OUT OF EACH END). AT ONE END THE POWER
WILL GO INTO THIS RESISTOR, AT THE OTHE
R ANOTHER TRACE WILL GO OFF TO SOME
OF THE OTHER ELECTRONICS ON THE BOARD.
WE WANT TO USE THE END THAT HAS THE
TRACES GOING TO OTHER CHIPS ON THE
BOARD. (CALL THIS POINT #1 (USE EITHER
RESISTOR - THERE ARE TWO)). POINT NUMBE
R TWO IS WHERE PIN 18 FROM THE APPLE
CONNECTOR (7 PINS DOWN FROM 25 ON THE
SAME SIDE) ENTERS ONTO THE PC BOARD
AND IMMEDIATELY GOES THROUGH TO THE
OTHER SIDE (AFTER ABT 1/2 "). THIS
IS POINT #2.  IF YOU TRACE WHERE TH
E THING COMES OUT ON THE OTHER SIDE,
```

YOU'LL FIND OUT THAT IT POPS BACK ON
THE SIDE IT STARTED FROM ABOUT 1/2"
LATER... THIS LITTLE LINK IS WHEERE WE
CUT THE TRACE TO INSERT THE SWITCH.
OK, WE CUT THE TRACE BETWEEN THE TWO
POINTS THAT IT GOES THROUGH THE PC
BOARD. LABEL THE OTHER PLACE WHERE THE
TRACE GOES THROUGH POINT#3. NOW WE
WILL ATTACH AN SPDT SWITCH TO THE BOARD
 SOLDER ONE WIRE TO POINT 3, AND ATTACH
IT TO THE CENTER TERMINAL OF THE SWITCH
THEN SOLDER A WIRE TO POINT 1 AND
ATTACH IT TO EITHER SIDE OF THE CENTER
SWITCH. LASTLY, TAKE A WIRE AND SOLDER
IT TO POINT 2 AND THEN TO THE UNUSED
PIN ON THE SWITCH. THERE YOU HAVE IT!
WHEN THE SWITCH HANDLE IS ON THE SAME
SIDE AS THE WIRE FROM POINT #1, REG-
ULAR OPERATION WILL TAKE PLACE. IF THE
SWITCH IS THROWN IN THE OTHER DIRECTION
THE CARD WILL BE WRITE PROTECTED.
(*PLEASE NOTE THAT THIS MODIFICATION
 WILL VOID YOUR WARRANTY AND THAT THE
USER ASSUMES AND WILL BE RESPONSIBLE
FOR ALL RISKS AND DAMAGES INCURRED IN
THE MAKING OR THE USE OF THIS MOD-
IFICATION, AND THAT THIS MODIFICATION
IS NOT GUARANTEED TO BE SUITABLE FOR
ANY PARTICULAR PURPOSE*)

COPY II PLUS 4.1 DISK-BACKUP INSTRUCTI
ONS
1/19/83

'T' INDICATES A TRACK NUMBER. WHEN A
RANGE OF TRACKS ARE TO BE COPIED, YOU
WILL SEE "TX-TXX". THIS MEANS SET THE
START TRACK TO "X", AND THE END TRACK
TO "XX". IF ONLY A SINGLE TRACK IS TO
BE COPIED, YOU WILL SEE "TX". THIS
MEANS USE "X" FOR BOTH START AND END
TRACKS. NUMBERS TO THE RIGHT ARE
PARAMETER CHANGES THAT SHOULD BE MADE
BEFORE COPYING THE TRACKS SHOWN.
"STEP" MEANS TRACK INCREMENT.

WHEN MAKING A BACKUP, BE SURE TO FOLLOW
THE STEPS IN ORDER. OFTEN A PARAMETER
WILL NOT BE RE-LISTED IF IT IS SET FOR
A PRIOR RANGE OF TRACKS.

IF A PARAMETER LISTING INCLUDES "SECTOR
EDIT", USE THE COPY II PLUS SECTOR EDIT
OR TO MODIFY THE TRACK AND SECTOR
SHOWN. BE SURE TO PATCH THE READ/WRITE
ROUTINES IF THE LISTING SHOWS "PATCHED"
AND TO USE THE CORRECT DOS (3.2 OR 3.3)

```
.

SOME DISKETTES CAN BE DUPLICATED USING
THE DEFAULT PARAMETERS (OR COPY DISK
FROM THE MAIN MENU).  IF THE DISKETTE
YOU WISH TO BACKUP IS NOT LISTED, TRY
THE DEFAULT SETTINGS OR COPY DISK FIRST
.

A "*" NEXT TO THE PRODUCT NAME IN-
DICATES THESE PARAMETERS WERE USER
SUBMITTED AND HAVE NOT BEEN VERIFIED BY
CENTRAL POINT SOFTWARE.  WE ENCOURAGE
OUR CUSTOMERS TO LET US KNOW WHEN THEY
BACKUP A DISK NOT ON THIS LIST.  THIS
INFORMATION IS MADE AVAILABLE TO ALL
COPY II PLUS OWNERS.

NOTE TO ALL IBM PC OWNERS: COPY II PC
IS NOW AVAILABLE.  SAME PRICE AS COPY
II PLUS FOR THE APPLE, AND IT IS HANDS-
DOWN THE FASTEST, MOST RELIABLE AND
MOST POWERFUL COPY PROGRAM FOR THE
PERSONAL COMPUTER.

ALIEN RAIN & TYPHOON    (BRODERBUND)
        T0-T5           9=0, 31=0, D=D5
, F=0
        T6-TE           E=DE

APPLE ADVENTURE *
        T0-T22          D=1, 10=96, 24=
96

APPLE LOGO *            (APPLE COMPUTER
)
        T0-T22
        T1              A=1, 4B=1, 50=1
  (ERROR 6 OK)

APPLE PANIC *           (BRODERBUND)
        T0-TD

APPLE WORLD *           (USA)
        T0-T23

APPLEWRITER II          (APPLE)
        T0-T22          10=96

APPLEWRITER ///         (APPLE)
        T0-T22          D=1, 10=96, 24=
96

A2-PB1  (PINBALL)       (SUB LOGIC)
        T0              10=96
        T1-T15          A=3, E=DB, F=AB
, 10=BF, 44=1,
                        45=D, 46=F
```

```
AZTEC *
        T0-22           D=1, 10=96, 24=
96

BACK-IT-UP II *         (SENSIBLE)
        T0              10=96, 9=0
        T1.5-TB.5       10=B5, A=3

BEER RUN
        T0              9=0
        T1.5-TD.5       D=1, 3B=40

CANNONBALL BLITZ *
        T0-T22
        T3-TF           3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

CANNONBALL BLITZ (ALTERNATE)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 17, SECTOR E.
          CHANGE ADDRESS CD FROM 49 TO
60

CASTLE WOLFENSTEIN      (MUSE)
        T0-T22          D=1, 31=0

CEILING ZERO *
        T0-T2
        T3-T11          9=0, E=D6, 1C=D
6, 34=1, 38=F9, 4F=1

CHESS 7.0 *             (ODESTA)
        T0-T22          10=96, 9=0

CHOPLIFTER & SERPENTINE (BRODERBUND)
        T0              A=3, 44=1, 45=D
, 9=0, 0=F, 50=3
        T1-T8           4=FD, 31=0, 43=
0, 45=10, 4F=1, 46=12
        T9              45=8, 46=D
        TA-TB           45=2
        TC-T1E.5 STEP .5        45=8, 1
0=D4, 51=1, D=1
        T20             45=6, D=0, 4F=0

NOTE: CHOPLIFTER, SERPENTINE, DAVID'S M
IDNIGHT MAGIC AND STARBLAZER USE TRACK A
RCING AND ARE VERY SENSITIVE TO DRIVE SP
EED.  IF YOU HAVE PROBLEMS, TRY REVERSIN
G DRIVES.

COLOSSAL CAVE ADVENTURE *
        T0-T22

CRANSTON MANOR          (ON-LINE)
```

```
        T0-T22
        T18            3B=1, A=1, 4B=1
, 4D=8, 50=1
                       (ERROR 6 OK)

CROSSFIRE              (ON-LINE)
        T0-TB          9=0
        T1             3B=1, A=1, 4B=1
, 4D=8, 50=1
                       (ERROR 6 OK)

CRUSH, CRUMBLE AND CHOMP *
        T0-T22         10=96, 9=0

DAVID'S MIDNIGHT MAGIC  (BRODERBUND)
        T0             A=3, 44=1, 45=D
, 9=0, 0=F, 50=3
        T1-TA          44=0
        TB             44=1, 31=0, 43=
0, 45=8
        TC-T19 STEP .5  10=F5, F=FD, 51
=1, 4F=1, D=1
        SEE NOTES FOR CHOPLIFTER

DB MASTER              (STONEWARE)
        T0-T5          10=96, 24=96, D
=1
        T6.5-T22.5     D=0

DEADLINE *             (INFOCOM)
        T0-T22

DESKTOP PLAN II        (VISICORP)
        T0-T22         10=96, 34=1, 36
=2A

DISK ORGANIZER *
        T0
        T1             3B=1, A=1, 4B=1
, 4D=8, 50=1
                       (ERROR 6 OK)
        T2-T4          D=1
        TA-TB

ELECTRIC DUET *        (INSOFT)
        USE COPY DISK FROM MAIN MENU

ESCAPE *
        T0-T22

EXECUTIVE SECRETARY *
        T0-T22         9=0, 8=1, 10=96

EXPEDITOR              (ON-LINE)
        T0-22          10=96
        T3 & T1F       3B=1, A=1, 4B=1
, 4D=8, 50=1
                       (ERROR 6 OK)
```

```
FORMAT II *
        USE COPY DISK FROM MAIN MENU

FS-1 (FLIGHT SIMULATOR) (SUB LOGIC)
        T0                      10=96
        T1.5-T21 STEP 1.5       E=DB, F
=AB, 10=BF, A=3, 4E=1
        T7-T8
        T9.5

GALACTIC GLADIATORS *
        T0-T20          10=B7, E=D7, 9=
0, 31=0
        T21-T22         34=1

GORGON                  (SIRIUS)
        T0              10=96, 9=0
        T1.5-E.5        D=1, 24=96, A=3
, E=DD, F=AD,
                        10=DA, 3B=40

HYPERSPACE WARS *       (CONTINENTAL)
        T0-T22          9=0

JAW BREAKER *           (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

KRELL LOGO *
        T0-T22

LIST HANDLER AND UTILITY *
        T1-T11
        T0              9=0, A=3, 44=1,
 45=D, 50=3
        T12-T22.5 STEP .5       D=1, E=
F5, F=D7, 10=F7
                                45=8, 4
6=D, 51=1
        (SEE NOTES FOR CHOPLIFTER)

MAGIC WINDOW *
        T0-T22

MICRO WAVE *            (CAVALIER)
        T0-T22
        T11             3B=1, A=1, 4B=1
, 4D=8, 50=1

MOUSKATTACK *           (SIERRA ON-LINE
)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 18, SECTOR 3
          CHANGE ADDRESS B1 FROM 49 TO
```

```
60

MULTI PLAN              (MICROSOFT)
        T0-T22          10=96

OLYMPIC DECATHALON *    (MICROSOFT)
        T0-T22          9=0

ORBITRON *
        T0-T1           9=0, 31=0
        T1.5-TF.5
        WRITE PROTECT COPY!

PFS & PFS REPORT        (SOFTWARE PUBLI
SHING CORP.)
        USE "COPY DISK" FROM MAIN MENU.
 AFTER COPYING AND BEFORE
        USING, PUT A TAB OVER THE WRITE
 PROTECT NOTCH OR  THE
        COPY WILL NOT WORK.

PHANTOMS FIVE           (SIRIUS)
        T0              9=0
        T2-T1C          3A=0, 50=20

PRISM *
        T0-T22

PRISONER *
        T0-T22

RASTER BLASTER          (OLD & NEW VERS
IONS - BUDGECO)
        T0              10=96
        T5-T11  STEP 4  D=1, 9=0, 31=0,
 A=2, E=AD,
                        F=DE, 3B=40
        T6-T12  STEP 4
        T7.5-TF.5  STEP 4
        T1.5-T3.5  STEP 2

SABATOGE *
        T0-T22
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

SARGON *                (HAYDEN)
        T0-T1A

SCREENWRITER II *
        COPY DISK, THEN SECTOR EDIT
          DOS 3.3 PATCHED
          TRACK 3, SECTOR B
          CHANGE ADDRESSES 94, 95, 96 T
O EA EA EA

SNOGGLE *               (BRODERBUND)
```

```
        T0-T9           9=0, 8=1

SPACE INVADERS *
        T0-T22          10=96

SNACK ATTACK            (DATA MOST)
        T0-T12
        SECTOR EDIT DOS 3.2 PATCHED
          TRACK 0 SECTOR 3
          CHANGE ADDRESS 63 FROM 38 TO
18

SNEAKERS                (SIRIUS)
        T0              9=0, 10=96, 44=
1, 45=10, D=1
        T1.5-TC.5       44=0
        TD.5            44=1

SOFTPORN ADVENTURE      (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

SPACE EGGS *            (SIRIUS)
        T0              9=0
        T2-T6
        T11-1A

SPACE VIKINGS *
        T0-T22

SPEED READING *
        T0-T22          9=0, 10=96

SPIDER RAID *           (INSOFT)
        T0
        T1-T17          A=3, E=92, F=93
, 4F=1, 10=95, 44=1
                        46=A, 9=0, 8=1,
 D=1, 24=96
                        3F=1, 34=1, 36=
2A, 37=97
                        31=0, 43=0
        T1.5-T17.5      E=95, 10=92
        (SEE NOTES FOR CHOPLIFTER)
        (ONLY WORKS ON NEW VERSIONS)

STARBLASTER *
        T0              10=96, 9=0
        T7-T20 STEP 1.5 E=DF, F=AD, 10=
DE

STARBLAZER              (BRODERBUND)
        SAME AS CHOPLIFTER

STARCROSS *             (INFOCOM)
        T0-T22          10=96
```

```
STELLAR INVADERS *      (APPLE)
        T0-T22

STOCK PORTFOLIO SYSTEM *
        T3-T22
        T0-T2           4=FD, 8=1, 10=A
D

TAX MANAGER *           (MICROLAB)
        USE COPY DISK FROM MAIN MENU

TAX PREPARER *          (HOWARDSOFT)
        USE COPY DISK FROM MAIN MENU

THRESHOLD               (ON-LINE)
        T0-T22
        T1-T23 STEP 22  3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

TUBE WAY *
        T0-T22

TYPING TUTOR *          (MICROSOFT)
        USE COPY DISK FROM MAIN MENU

ULTIMA II *
        COPY DISK, THEN SECTOR EDIT
          TRACK 3, SECTOR 0C
          CHANGE ADDRESSES 84, 85, 86 A
LL TO EA.

ULTIMA II *
        T0-T22          10=96, 9=0, 34=
1, 31=0

VERSAFORM *
        T0-T22

VISICALC                (VISICORP)
        T0-T16

VISICALC ///            (APPLE COMPUTER
)
        T0-T22          10=96, 24=96, D
=1

VISIDEX, VISISCHEDULE, VISITERM, VISITR
END/VISIPLOT  (VISICORP)
        DON'T USE BIT COPY.  USE "COPY
DISK" FROM MAIN MENU.

VISIFILE                (VISICORP)
        T0-T22          10=96, 34=1, 36
=2A, 37=EB, 3E=2

WIZARDRY * (FRONT SIDE)
```

```
        COPY DISK THEN USE BIT COPY:
        T3-T23          10=96, 24=96, D
=1
        NOTE: WRITE PROTECT BACKUP OR I
T WILL NOT WORK.

WIZARDRY * (BACK SIDE)
        T1-T22          10=96, 24=96, D
=1

WORD HANDLER *
        USE COPY DISK FROM MAIN MENU

ZARGS *                 (INOSFT)
        SAME AS SPIDER RAID

(?=MENU, 1-9) ->:

---------------------------------------
       COPY II+ PLUS PARAMETERS
---------------------------------------

FROM:
     THE FORBIDDEN ZONE
     THE ROM RAIDER

UPDATE:3/28/83

3-D Graphics System * (Cal Pacific Computer)
     T0-T8
     T11-T12
     T15-T17

3-D Graphics System *
     T0-T2
     T4-T8
     T11-T18

Adventure To Atlantis * (Synergistic)
     T0-T22             10=96 24=96 9=0 31=0 D=1

Air Simulator *         (Mind Systems)
     T0-TF

Air Traffic Controller *
     T0-T22             10=96
     T23                31=0 50=1 10=96

Akalabeth *
     T0                 9=0 31=0
     T2-T3              E=DE F=AA 10=AD
     T6-T18

Alien Rain & Typhoon     Broderbund)
     T0=T5              9=0 31=0 D=D5 F=0
     T6-TE              E=DE

Alkem Stones *
```

```
     T0-T22             A=3 10=96

Apple Adventure *
     T0-T22             D=1 10=96 24=96

Apple Cillin II *
     T0-TC

Apple //e  Business Graphics *
     T0-T22             D=1 10=96 24=96

Apple /// Business Graphics *
     T0-T22  (error 2 OK)

Apple Logo *
     T0-T22
     T1                 A=1 4B=1 50=1 E=FC 19=FD 1C=AA 1F=EE
            or for T1
     T1                 A=1 4B=1 50=1 E=AA 1C=AA
               or
     T1                 A=1 4B=1 50=1 3B=1 4D=8

NOTE:  We have been told that Apple Logo requires persistance!
Keep trying
track 1 until the disk works.

Apple Panic *           (Broderbund)
     T0-TD

Apple Panic *
     T0-T5              9=0 F=0
     T6-TD              E=DE

Apple Pilot and Super Pilot *
     T0-T22

Apple World *           (USA)
     T0-T23

Apple Writer II (and IIe)
     T0-T22             10=96

Apple Writer II Pre-Boot *
     T0-T22             10=96 9=0

Apple Writer ///        (Apple)
     T0-T22             D=1 10=96 24=96

A2-PB1 (Pinball)        (Sub-Logic)
     T0                 10=96
     T1-T15             A=3 E=DB F=AB 10=BF 44=1 45=D 46=F

AZTEC *
     T0-T22             D=1 10=96 24=96

Back-It-Up II *         (Sensible)
     T0                 10=96 9=0
     T1.5-TB.5          10=b5 A=3
```

```
Battle of Shilo *
     T0-T22             E=D4 10=b7

Beer Run
     T0                 9=0
     T1.5-TD.5          D=1 3B=40

Bomb Alley *
     T0-T22             E=D4 10=B7 34=1 37=6E 38=fe

Borg *                  (Sirius)
     T0                 10=96 9=0
     T1.5-TB.5          D=1 24=96 A=3 E=DD F=AD 10=DA 3B=40
     TD-T20

Cannonball Blitz *
     T0-T22
     T3-TF              3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

Cannonball Blitz (alternate)
     T0-T22             10=96
     Sector Edit Dos 3.3 Patched
       Track 17, Sector E
       Change Address CD From 49 to 60

Castle Wolfenstein      (Muse)
     T0-T22             D=1 31=0

Caves of Olympus *
     T0-T22             10=96 9=0

Ceiling Zero *
     T0-T2
     T3-T11             9=0 E=D6 1C=D6 34=1 38=F9 4F=1

Chess 7.0 *             (Odesta)
     T0-T22             10=96 9=0

Chess 7.0 *
     T0-T22             10=96 9=0 8=1 3E=2

Choplifter, Serpentine, & Starblazer (Broderbund)
     T0                 A=3 44=1 45=d 9=0 0=F 50=3
     T1-T8              4=FD 31=0 43=0 45=10 4F=1 46=12
     T9                 45=8 46=D
     TA-TB              45=2
     TC-T1E.5 Step .5   45=8 10=D4 51=1 D=1
     T20                45=6 D=0 4F=0

NOTE:  Choplifter, Serpentine, David's Midnight Magic and
Starblazer use track
arcing and are very sensitive to drive speed. If you have
problems, try
reversing drives.
       -- Sea Fox may also copy using these parameters --

Colossal Cave Adventure *
```

```
     T0-T22

Congo *
     T0-T22              D=1 9=0 24=96 10=96

Cranson Manor            (On-Line)
     T0-T22
     T18                 3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

Crossfire                (On-Line)
     T0-T8               9=0
     T1                  3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

If you want CRUSH, CRUMBLE AND CHOMP you're crazy! That game
sucks!

Dark Crystal             (On-Line)
     Copy all 4 sides from main menu
     Sector Edit side 1A as follows:
       Track 5, Sector F change address A8-AA all to EA
       Track 7, Sec C, change addressess 22-24 all to EA

David's Midnight Magic  (Broderbund)
        T0               A=3, 44=1, 45=D, 9=0, 0=F, 50=3
        T1-TA            44=0
        TB               44=1, 31=0, 43=0, 45=8
        TC-T19 STEP .5  10=F5, F=FD, 51=1, 4F=1, D=1
        SEE NOTES FOR CHOPLIFTER

DB MASTER                (STONEWARE)
        T0-T5            10=96, 24=96, D=1
        T6.5-T22.5       D=0

DEADLINE *               (INFOCOM)
        T0-T22

DESKTOP PLAN II          (VISICORP)
        T0-T22           10=96, 34=1, 36=2A

DISK ORGANIZER *
        T0
        T1               3B=1, A=1, 4B=1, 4D=8, 50=1
                         (ERROR 6 OK)
        T2-T4            D=1
        TA-TB

DLM Software *
        T0-T22

Dragon Fire *
        T0-T22           10=96 9=0

Early Games *
        Use Copy Disk from Main Menu

Education Activities Software *
        T0-T22
```

```
Einstein Computer *
        Copy Disk from Main Menu
        Sector Edit Track 8, Sector 4
        Change Addresses 2A-2C from BD 8C C0 to 4C E2 91

ELECTRIC DUET *         (INSOFT)
        USE COPY DISK FROM MAIN MENU

ESCAPE *
        T0-T22

EXECUTIVE SECRETARY *
        T0-T22          9=0, 8=1, 10=96

EXPEDITOR               (ON-LINE)
        T0-22           10=96
        T3 & T1F        3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

First Class Mail *
        Use Copy Disk from Main Menu

FORMAT II *
        USE COPY DISK FROM MAIN MENU

FS-1 (FLIGHT SIMULATOR) (SUB LOGIC)
        T0                      10=96
        T1.5-T21 STEP 1.5       E=DB, F=AB, 10=BF, A=3, 4E=1
        T7-T8
        T9.5

GALACTIC GLADIATORS *
        T0-T20          10=B7, E=D7, 9=0, 31=0
        T21-T22         34=1

GORGON                  (SIRIUS)
        T0              10=96, 9=0
        T1.5-E.5        D=1, 24=96, A=3, E=DD, F=AD,
                        10=DA, 3B=40

HYPERSPACE WARS *       (CONTINENTAL)
        T0-T22          9=0

JAW BREAKER *           (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

KRELL LOGO *
        T0-T22

LIST HANDLER AND UTILITY *
        T1-T11
        T0              9=0, A=3, 44=1, 45=D, 50=3
        T12-T22.5 STEP .5       D=1, E=F5, F=D7, 10=F7
                                45=8, 46=D, 51=1
        (SEE NOTES FOR CHOPLIFTER)
```

```
MAGIC WINDOW *
        T0-T22

MICRO WAVE *            (CAVALIER)
        T0-T22
        T11             3B=1, A=1, 4B=1, 4D=8, 50=1

MOUSKATTACK *           (SIERRA ON-LINE)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 18, SECTOR 3
          CHANGE ADDRESS B1 FROM 49 TO 60

MULTI PLAN              (MICROSOFT)
        T0-T22          10=96

OLYMPIC DECATHALON *    (MICROSOFT)
        T0-T22          9=0

ORBITRON *
        T0-T1           9=0, 31=0
        T1.5-TF.5
        WRITE PROTECT COPY!

PFS & PFS REPORT        (SOFTWARE PUBLISHING CORP.)
        USE "COPY DISK" FROM MAIN MENU. AFTER COPYING AND BEFORE
        USING, PUT A TAB OVER THE WRITE PROTECT NOTCH OR  THE
        COPY WILL NOT WORK.

PHANTOMS FIVE           (SIRIUS)
        T0              9=0
        T2-T1C          3A=0, 50=20

PRISM *
        T0-T22

PRISONER *
        T0-T22

RASTER BLASTER          (OLD & NEW VERSIONS - BUDGECO)
        T0              10=96
        T5-T11  STEP 4  D=1, 9=0, 31=0, A=2, E=AD,
                        F=DE, 3B=40
        T6-T12  STEP 4
        T7.5-TF.5  STEP 4
        T1.5-T3.5  STEP 2

SABATOGE *
        T0-T22
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

SARGON *                (HAYDEN)
        T0-T1A

SCREENWRITER II *
        COPY DISK, THEN SECTOR EDIT
          DOS 3.3 PATCHED
          TRACK 3, SECTOR B
          CHANGE ADDRESSES 94, 95, 96 TO EA EA EA

SNOGGLE *               (BRODERBUND)
        T0-T9           9=0, 8=1

SPACE INVADERS *
        T0-T22          10=96

SNACK ATTACK            (DATA MOST)
        T0-T12
        SECTOR EDIT DOS 3.2 PATCHED
          TRACK 0 SECTOR 3
          CHANGE ADDRESS 63 FROM 38 TO 18

SNEAKERS                (SIRIUS)
        T0              9=0, 10=96, 44=1, 45=10, D=1
        T1.5-TC.5       44=0
        TD.5            44=1

SOFTPORN ADVENTURE      (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

SPACE EGGS *            (SIRIUS)
        T0              9=0
        T2-T6
        T11-1A

SPACE VIKINGS *
        T0-T22

SPEED READING *
        T0-T22          9=0, 10=96

SPIDER RAID *           (INSOFT)
        T0
        T1-T17          A=3, E=92, F=93, 4F=1, 10=95, 44=1
                        46=A, 9=0, 8=1, D=1, 24=96
                        3F=1, 34=1, 36=2A, 37=97
                        31=0, 43=0
        T1.5-T17.5      E=95, 10=92
        (SEE NOTES FOR CHOPLIFTER)
        (ONLY WORKS ON NEW VERSIONS)

STARBLASTER *
        T0              10=96, 9=0
        T7-T20 STEP 1.5 E=DF, F=AD, 10=DE

STARBLAZER              (BRODERBUND)
        SAME AS CHOPLIFTER

STARCROSS *             (INFOCOM)
        T0-T22          10=96

STELLAR INVADERS *      (APPLE)
        T0-T22
```

```
STOCK PORTFOLIO SYSTEM *
        T3-T22
        T0-T2           4=FD, 8=1, 10=AD

TAX MANAGER *           (MICROLAB)
        USE COPY DISK FROM MAIN MENU

TAX PREPARER *          (HOWARDSOFT)
        USE COPY DISK FROM MAIN MENU

THRESHOLD               (ON-LINE)
        T0-T22
        T1-T23 STEP 22  3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

TUBE WAY *
        T0-T22

TYPING TUTOR *          (MICROSOFT)
        USE COPY DISK FROM MAIN MENU

ULTIMA II *
        COPY DISK, THEN SECTOR EDIT
          TRACK 3, SECTOR 0C
          CHANGE ADDRESSES 84, 85, 86 ALL TO EA.

ULTIMA II *
        T0-T22          10=96, 9=0, 34=1, 31=0

VERSAFORM *
        T0-T22

VISICALC                (VISICORP)
        T0-T16

VISICALC ///            (APPLE COMPUTER)
        T0-T22          10=96, 24=96, D=1

VISIDEX, VISISCHEDULE, VISITERM, VISITREND/VISIPLOT  (VISICORP)
        DON'T USE BIT COPY.  USE "COPY DISK" FROM MAIN MENU.

VISIFILE                (VISICORP)
        T0-T22          10=96, 34=1, 36=2A, 37=EB, 3E=2

WIZARDRY * (FRONT SIDE)
        COPY DISK THEN USE BIT COPY:
        T3-T23          10=96, 24=96, D=1
        NOTE: WRITE PROTECT BACKUP OR IT WILL NOT WORK.

WIZARDRY * (BACK SIDE)
        T1-T22          10=96, 24=96, D=1

WORD HANDLER *
        USE COPY DISK FROM MAIN MENU

ZARGS *                 (INOSFT)
        SAME AS SPIDER RAID
```

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

STARTING IN 86,SEARS WILL ISSUE THE "DISCOVER" CREDIT CARDS.THESE SHOULD SOON PROVE BETTER THAN VISA AND MASTER CARD WITH CONSUMER AND DEALER DISCOUNTS AND 35 BILLION IN CREDIT AVAILABLE.
 REMEMBER THE CHAOS WHEN VISA MAILED OUT MILLIONS OF CREDIT CARDS! WATCH YOUR MAILBOX (AND YOUR NEIGHBORS) IN 86 FOR YOUR VERY OWN UNUSED "DISCOVER" CREDIT CARD(S).

NUFF SAID-
BOOTLEG

```
        ======================
        DOS Notes & Pointers..
        ======================

        -=> By THE FREEZE <=-

Suggested reading: Beneath Apple DOS
                   The DOS Manual

Needed equipment: Apple, Drive
                  IQ > 60
```

I will start this article by giving you an overview of what is in DOS and what it does. First of all there is the RWTS. This allows you to read or write a sector at a time. All operations are done either directly or indirectly through this. Starting at $B600 and ending at $BFFF RWTS takes up about 2.5K. Next is the File Manager. This goes from $AAC9 to $B5FF This is a bunch of subroutines which execute your commands from basic.
Then there are the main DOS routines. These interpret your commands and tell the file manager what to do, which in turn uses the RWTS to do them. These routines go from $9D00 to $AAC8.
When you have MAXFILES set to 3, DOS reserves memory from $9600 to $9CFF. Setting MAXFILES higher will take up more memory, lower than $9600.
There is another part of DOS, which resides in the latter part of page 3 or from $3D0 to $3FF. This is called the Dos Vector Table. I will go into detail on that later.

Well now, lets say you put a disk in

the drive and turned your computer on.
Then you loaded a file, edited it and
saved it. Why don't we take a look and
see exactly what is happening.

When you turn your computer on (if
you have autostart) the code on your
drive controller prom takes over.
This loads in a routine at $800.
This is called Boot 0. Then it jumps
to $801 and executes that code (boot
1). That code loads in sectors 1
through 9 which in turn loads in the
rest of DOS. Then it looks to see if
you have a HELLO program and jumps
to it. The first thing it does when
loading in a program, in this case
the HELLO program, is look at the
catalog track. Then after it finds
the file and the track and sector it
starts on, it reads in the first
sector. The first sector of a program
is called the Track Sector List or
TSL. This is a listing of all tracks
and sectors that have data for that
program. DOS reads this into memory
and then starts loading the program
in. But where does it know where to
load the program in and how does it
know what file type it is?
The file type was back on the catalog,
more (lots more) on that later...
On the first sector of data, not the
TSL, in the first two bytes is the
address to start loading in at. These
bytes as usual are in reverse order.
Well, now you know a little of how
DOS works. Lets go into more detail.

Here is where I will probably lose
you. If it gets confusing hang on.
Now we will look at track $11, which
is the catalog track. The VTOC or
Volume Table Of Contents is stored
at track $11, sector $00. This tells
DOS such things as: what sectors are
free, volume #, DOS version, first
link to catalog sector...
Bytes $01 & $02 of the VTOC tells us
where to find the
first catalog sector. This usually is
track $11, sector $0F. Byte $02 is
the DOS version. Either a "1", "2",
"3", for DOS 3.1, 3.2, 3.3, consec-
utively. Byte $07 is the volume #
usually $FE (254). The next thing of
interest is the Bit Map. Starting at
byte $38 you will see "FFFF0000".

For now, ignore the last two bytes.
The "FFFF" is a binary representation
of what sectors are free on a certain
track. In the two bytes there are
16 bits. Makes sense doesn't it, 16
bits and 16 sectors. If the bit is
set or a "1" then that sector is free.
If it is a "0" then it's used.
Now lets look at the catalog link.
On track $11, sector $0F, byte $01,
are two bytes that tell what track
and sector to find the first catalog
sector. This is almost always track 11
sector $0F. On track $11, sector $0F,
bytes 1 and 2, is a pointer to the
next sector, track $11, sector $0E.
The links continue until sector $01
where you will see zero's in those
bytes. I have been asked many times
how to get more than 105 files onto
a disk. If you edit the second and
third bytes on track $11, sector $01
to "100F", you will be able to use
track $10, sector $0F for a catalog
sector. You can continue on track
10 sector F and make a link to the
next sector and so on. Be sure to
mark it on the bit map or it will
get wiped out when DOS has to write
there. Well, we have covered most
of the VTOC, lets look at how the
catalog sectors are formatted.

Starting at byte $0B on any catalog
sector, is the entry for a file.
The first two bytes after that, tells
what track and sector the program
starts on. Then is the file type (more
on that later). Next comes the file
name, up to 30 characters. The last
byte before the next entry tells us
how many sectors the file takes.
This usually never goes over 255
sectors, however text files can take
more than 255 sectors. Now we can
look at the file type. We have to look
at this at the binary level. If the
first bit is set, it is a text file.
If the second is set, it's Integer.
Third is applesoft, fourth is binary.
If the eigth or MSB is set, the file
is locked. It's really quite simple.
$00 means a text file. $80 means a
locked text file. If it is a $84,
we have a locked binary file.

Now for the complicated stuff, how
DOS writes sectors, INITs a disk,

the "6 & 2" split. Lets say you put a
blank disk in the drive, initialized
it, and saved a file onto it. Lets see
what happens. First off, at $A54F is
the INIT routine. If you did A54FG
from monitor, it would INIT your disk
without a hello program. This lets
your disk boot faster because it does
not have to load in that file.
Ok. So you type in "INIT HELLO". DOS
takes over and starts formatting your
disk starting with track 0 and ending
with track $23. Then it writes the
catalog track and VTOC. Last it writes
in DOS. Lets take a close look at a
disk at the track level. First off we
have what is called a GAP. This is made
up of "FF"'s. Then we have the prolouge
marks, ye olde "D5 AA 96". After that
comes the volume, track, sector, check-
sum, epilouge "DE AA EB". Then comes a
smaller GAP with a different prolouge
"D5 AA AD". Then $342 bytes of user
data. Oops! $342 bytes of user data?
I thought there were only $FF or 255
bytes per sector! (more on this later).
Then we have the checksum. And last
we have the epilouge "DE AA EB".
There are certain bytes that DOS
doesn't write as data. These bytes are
used in proulouge and epilouge marks.
DOS looks for these when trying to find
a sector. Now for the "6 & 2" split.
The hardware on the apple doesn't allow
for more than $3F different bytes to
be written. That's why they used the
"5 & 3" split on 3.2 disks. What that
means is that from one byte, five bits
are taken out and form one byte. The
other three bits form one byte also.
The six and two split is the same thing
as the five and three but allows for
more combinations.

Now for a little on copy protection.
Back in the good 'ol days we could just
demuffin everything. All they had was
a modified DOS or changed address marks
etc. After that they got a little
smarter and some wrote their own DOS
or used a modified RWTS. But nothing
stops us pirates, all you had to do
is read in data through their RWTS and
write it back out standard. Then they
got dirty, using the text page and the
input buffer for data or code. They
even used the stack (page 1) for code.
To get around this, NMI card like

crackshot and cracking chips were made. these dumped all memory to disk allowing the text page and the input buffer to be undisturbed. The newest thing seems to be SPIRAL TRACKING. The first game I saw this on was Maze Craze. Cracking it was quite easy though. All you had to do is cut out one part of disk access (at $855) that wasn't even needed. But who knows what we will be up against in the future.

I suggest you read "Beneath Apple DOS" and look at the DOS manual supplied with your Apple. This is for beginners or people who are too lazy to read a book...

I would appreciate lots of questions, I may not have made myself too clear or you may want to know more about a certain area. Just leave me e-mail.

The Freeze

D O S   T R I C K S

TRY THIS TO SEE ANY DOS, REMOVE THE REAR MOST SET OF RAM CHIPS FROM YOUR APPLE (THE ONES NEAR THE I/O SLOTS). THEN INIT A DISK, REPLACE THE RAM AND BOOT UP UNDER THE PROGRAM YOU WISH TO DEPROTECT. THEN FORCE A REBOOT WITH THE DISK YOU INITED IN DRIVE 1. THE DOS FROM THE PROTECTED DISK WILL (IN MOST CASES) STILL BE IN THE RAM UP TOP....

THIS NEW DOS IS A SLAVE AT 32 K AND THE OLD (AND PROTECTED DOS) IS STILL AT 48 K. THIS WILL WORK ON ABOUT 50% OF THE PROGRAMS. ENJOY

YOU CAN ALSO REMOVE THE TOP 32K AND GET TWICE AS MUCH.

=======================================

CHECKSUM TRICK

A VERY HANDY TECHNIQUE FOR TAKING A LOOK AT THE DATA ON A PROTECTED DISK IS TO DISABLE THE CHECKSUM IN THE RWTS. THE FORMATS OF MANY PROTECTED DISKS VARY ONLY IN THIS CHECKSUM, SO TURNING IT OFF SHOULD ALLOW ANY STANDARD TRACK/SECTOR UTILITY TO LOOK AT THE DISK! TO DO THIS, BOOT UP THE DOS THAT YOU WISH TO USE, AND ENTER THE MONITOR.

THEN ENTER B942:18 FOR DOS 3.3 OR B963:18 FOR DOS 3.2. THIS CHANGES A SET CARRY INSTRUCTION TO A CLEAR CARRY INSTRUCTION. NOW RETURN TO DOS AND RUN YOUR EDITOR. IF THE DISK YOU ARE LOOKING AT IS PROTECTED WITH THIS SYSTEM, YOU SHOULD BE ABLE TO READ IT NOW. TO MAKE THIS CHANGE TO A DOS ON A DISK, THIS DATA IS CONTAINED IN TRACK 0 SECTOR 3, AT EITHER BYTE $42 OR BYTE $63, FOR DOS 3.3 OR 3.2, RESPECTIVELY.
GOOD LUCK.......
RANDY

=======================================

TO AVOID RE-LOADING THE LANGUAGE CARD
ON BOOTUP (A MAJOR IRRITATION)
CHANGE THE FOLLOWING :

IN A 48K SYSTEM, CHANGE $BFCC TO 00 AND
$BFCF TO 00 : THIS WILL PREVENT THE
LANGUAGE CARD FROM BEING WRITTEN TO.
(INITIALIZE A DISKETTE WITH THIS DO
TO MAKE IT BOOT UP IN THIS FASHION)

(IF YOU LOOK AT THE CODE, YOU CAN MAKE
THE SAME MODS IN A COPY OF A SYST
MASTER ON THE DISK ITSELF, SO A MASTER
CREATE WILL PUT THIS DOS ON A DISKETTE.
CHANGE THE CODE THAT SAYS LDA C081 WITH
LDA C000 -- THAT SSHOULD WORK FINE.

=======================================

GET INTO MONITOR FROM A NORMAL DISK.

```
TYPE:   400<A800.ABFFM
```

POOF THERE YOU HAVE ALL THE DOS COMANDS
NOTICE THAT ALL THE LETTERS IN THE
COMAND ARE FLASHING BUT THE LAST ONE
THAT IS TO TELL YOU WHERE THE COMAND
ENDS. NOW NOTICE WHERE THE INIT,LOAD,
BLOAD,SAVE,BSAVE,CATALOG, ETC...
THEN BOOT SOMETHING LIKE BRAIN SURGEON
OR SOMETHING THAT HAS SOMETHING LIKE
A NORMAL FORMAT THEN TYPE THAT LINE
AND THEN YOU CAN SEE IF THEY CHANGED
ANY OF THE COMANDS!

=======================================

E.D.D. PARMAMETERS

```
A2-FS1:
   T0 - T6   INC 1.5
   T7 - T8
   T9.5-TA.5
   TC - T21  INC 1.5
ABM:NORM
A.E SIDE A:
   T1.5-TD.5
   TE -T18.5 INC 1.5
    SIDE B:NORM
ASCII EXPRESS PROFESSIONAL:NORM
ADVENTURE:NORM
AIRSIM-1:NORM
 WRITE-PROTECT BEFORE BOOTING!
ALGEBRA 1:NORM
ALKEMSTONE:NORM
APPLE PRESENTS- ERNIE'S QUIZ:NORM
APPLE PRESENTS- INSTANT ZOO:NORM
APPLE PRESENTS- SPOTLIGHT:NORM
APPLE PRESENTS- MIX AND MATCH:NORM
APPLE WORLD:
   T0-T23
APPLE WRITER:NORM
APPLE WRITER II:NORM
APPLE WRITER IIE:NORM
APPLE WRITER 80 COLMN PRE-BOOT:NORM
APVENTURE TO ATLANTIS:NORM
ARCADE MACHINE:
   T0 -T11
   T12.25-T21.25
ASTEROID FIELD:NORM
AUDEX:NORM
AZTEC:NORM
BANK STREET WRITER:
   T0 -T1A
   T1B-T22 PPM#3 OR #4
BATTLE FOR NORMANDY:SEE MINER 2049ER
BEER RUN:
   T0 PARM 28=2 OR 3
   T1.5-TD.5  PPM#2
BENEATH APPLE MANOR:(SPECIAL EDITION)
   T0-T22  PARM 0=3
BILL BUDGE 3-D GRAPHICS:NORM
BILL BUDGE SPACE ALBUM:NORM
BILL BUDGE TRILOGY OF GAMES:NORM
BORG:
   T1.5-TB.5
   TD-TE
   T0 PARM 28=2 OR 3
BUG ATTACK:
   T0-T22
   T1.5 =PPM#2
   T22 =PPM#2
BUSINESS GRAPHICS:NORM
CAMPAIGN TRILOGY:NORM
CANNONBALL BLITZ:NORM
CANYON CLIMBER:NORM
CARTELS AND CUTTHROATS:NORM
CASTLE WOLFENSTEIN:NORM
CCA DATA MANAGEMENT:NORM
```

```
CHECKERS (ODESTA):NORM (T0-T6)
CHESS 7.0 (ODESTA):NORM
CHOPLIFTER:
 NOTE:SOMETIMES VERY HARD TO COPY
   T0-TB PARM 28=2  00=3
   TC.25-T21.25
   T22
COMPUTER AMBUSH:NORM
COMPUTER AMBUSH VER 2:NORM
COMPUTER BISMARK:NORM
CONGO:NORM
COPTS & ROBBERS:SEE EPOCH
COPY II PLUS:NORM
CRANSTON MANOR:
   T0-T22
   T18 PPM#3
CRIME WAVE:NORM (T0-T11)
CRISIS MOUNTAIN:NORM
CRITICAL MASS:
 SIDE A:
   T0-TA
   T22  PPM#3
 SIDE B:NORM
CROSSFIRE:
   T0-T22
   T1 PPM#3
CROSSWORD MAGIC (BOTH SIDES):
   T0-T22 PPM#2
CUSTOM MICRO SYSTEMS ASSEMBLER:
   T0-T23:NORM
D.B. MASTER AND UTILITIES:
   T0 - T5
   T6.5-T22.5
DARK CRYSTAL:NORM
DATA TREE:NORM
DEADLINE:NORM
DESKTOP PLAN II:NORM
DISK EDIT 2.0 (DISK EDITOR):
   T0
   T1.5 -T5.5
   T21.25-T22.25
DISK RECOVERY:NORM
 IF THAT DOESN'T WORK, TRY:
   T0
   T1.25-T10.25  PPM#2
DOS ENHANCER:NORM
DUNG BEETLES:NORM
EASY-WRITER:NORM
EDU-PAINT:NORM
EINSTEIN COMPILER:NORM
ELECTRIC DUET:NORM
EMPIRE I: WORLD BUILDERS:NORM
EPOCH:
   T0 PARM 28=2 OR 3
   T1.5-TF.5 PPM#2
EVOLUTION:
   T0.25-T18.25
E-Z DRAW:NORM
```

```
FINANCIAL MANAGMENT SYSTEM III:
   T0-22
   T3 PARM 4=10 9=3 A=14 B=13 11=3;
      PPM#3 OR #4
FIRE BUG:NORM
GALACTIC EMPIRE:NORM
GALACTIC REVOLUTION:NORM
GAMMA GOBLINS:SEE BEER RUN
GAME SHOW & SUBJECTS:NORM
GERMAN/ENGLISH HANGMAN:NORM
GERMANY 1985:NORM
GORGON:
   T0  PARM 28=2 OR 3
   T1.5-TE.5  PPM#2
HADRON:SEE GORGON
HAIL:NORM
HEAD-ON:NORM
HELLFIRE WARRIOR:NORM
HOME ACCOUNTANT:NORM
INFORMATION MASTER:NORM
JAWBREAKER:
   T0-T22
   T3 PPM#3
KNIGHT OF DIAMONDS:PPM#2
L.A. LAND MONOPOLY:NORM
LABYRINTH:SEE CHOPLIFTER
LETTER PERFECT:NORM
LINGUIST:NORM
LIST HANDLER & UTILITIES:
   T11
   T12.25-T22.25  PARM 0=3
   T0  PARM 0=0 28=2
MASTER TYPE:NORM
MATH GAMES:NORM
MERLIN ASSEMBLER:NORM
MICROBE:NORM
MIDNIGHT MAGIC:
   T0 - T12
   T13.25-T15.25
   T22
MINER 2049ER:
   T1-T22
   T0  PPM#3 OR #4
MINGS CHALLENGE:SEE MINER 2049ER
MISSION ASTEROID:NORM
MOPTOWN:
   T0-T22  PARM 28=3
MILTIPLAN:
   T0-T22
   TA PPM#3 OR #4
MUSICOMP:NORM
NIBBLES AWAY II:NORM
NIGHT MISSION PINBALL:NORM
ODYSSEY:NORM
OLYMPIC DECATHALON:NORM
OLYMPIC INSURANCE SYSTEMS:NORM
PEGASUS II:SEE JAWBREAKER
PFM:NORM
```

```
PFS-FILE:
   T1-T22
   T0  PPM#3 OR #4
 >> WRITE-PROTECT BEFORE BOOTING !!! <<
PFS-FILE IIE:SEE PFS-FILE
PFS-GRAPH:SEE PFS-FILE
PFS-REPORT:SEE PFS FILE
PHANTOMS FIVE:SEE EPOCH
PINBALL CONSTRUCTION SET:NORM
POOL 1.5:PPM#2
PRESIDENT ELECT:NORM
PRISONER:NORM
PULSAR II:
   T: - T19
   T1A.5-T1D.5
QUEEN OF PHOBOS:NORM (T0-T1A)
REAR GUARD:NORM
RENDEZVOUS:SEE MINER 2049ER
RESCUE AT RIGEL:NORM
ROBOTWAR:NORM
SABATOGE:SEE JAWBREAKER
SARGON II:NORM
SCREENWRITER II:NORM
SEA FOX:SEE CHOPLIFTER
SENSIBLE SPELLER:NORM
SERIES RU-2:NORM
SERIES SP-2:NORM
SERIES FR-2:NORM
SERPENTINE:SEE CHOPLIFTER
SNEAKERS:SEE BEER RUN
SOFTPORN ADVENTURE:SEE JAWBREAKER
SORCEROR OF SIVA:NORM
SPACE EGGS:NORM
SPANISH/ENGLISH HANGMAN:NORM
SPECTRE:NORM
SPITFIRE SIMULATOR:NORM
SPY'S DEMISE:NORM
STARCROSS:NORM
STAR THIEF:
   T0-T13
   T22  PPM#3
SWASHBUCKLER:PARM 28=10
SUPER DISK COPY III:NORM
TWALA'S LAST REDOUT:PPM #2
TAXMAN:NORM
TEMPLE OF APSHAI:NORM
TERRORIST:NORM
 IF THAT DOESN'T WORK, TRY:
   T0-T1F
   T20.75-T22.75
THE ROUTINE MACHINE:NORM
THIEF:
   T0-T22
   T4-T5  PPM#2
THREE MILE ISLAND:NORM
THRESHHOLD:SEE CROSSFIRE
THUNDER BOMB:NORM (T0-T11)
TIC TAC SHOW:
   T0
   T1.5-T4.5
   T6-T22
  SERIES DISKS:NORM
TIME ZONE SIDE A:SEE MINER 2049ER
          SIDES B-L:NORM
TORPEDO FIRE:NORM
TRACK ATTACK:SEE CHOPLIFTER
TRANSEND: PPM#2
TRANSYLVANIA:NORM
TUBEWAY:NORM
TYPING TUTOR:NORM
ULTIMA:NORM
ULTIMA II:SEE RENDEZVOUS
ULYSSES:NORM
VISICALC 3.3:NORM
VISICALC 80 COLMN PRE-BOOT:NORM
VISICALC IIE:NORM
VISIDEX:NORM
VISIFILE:NORM
VISIPLOT:NORM
VISISCHEDULE:NORM
VISITERM:NORM
VOCABULARY BUILDER-FRENCH:NORM
VOCABULARY BUILDER-GERMAN:NORM
VOCABULARY BUILDER-SPANISH:NORM
WARP FACTOR:PPM#2
WIZARDRY:PPM#2
WORD HANDLER:
 NOTE: SOMETIMES VERY HARD TO COPY
   T11
   TB.25-T10.25
   T0-TA PARM 0=0 28=2
WORD RACE:PPM#2
WORLDS GREATEST BLACK-JACK:NORM
ZENITH:SEE CHOPLIFTER
ZOOM GRAPHICS:NORM
ZORK I:NORM
ZORK II:NORM
ZORK III:NORM
----------------------------------------

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

CBS IN NYC HAS AN EXPERIMENTAL RESEARCH
STATION GOIN AT 149.195,149.220 AND
149.245 MHZ TO DETERMINE THE FEASIBILIT
Y OF MOBILE SATELITE USE.

OTHER RESEARCH STATION ARE-

MOTOROLA-SCHAUMBERG,ILL. 1359.6 AND
1360.06 MHZ.

MOTOROLA-CANTON,MASS. SAME AS ABOVE

MOTOROLA-CUPERTINO,CA. SAME AS ABOVE
```

NUFF SAID-
BOOTLEG
       ATTENTION][E hackers! Having problems doing serious disk snooping because you can't reset to the monitor? For just 19.95 + 2.00 s/h I will send you the chip to make it all possible. Easily installed in 5 minutes with no cutting or soldering. These chips come programmed, tested, and ready to go. Cracking hints, tips and docs. included. Send check or money order to: Hacker Chips Inc. P.O. Box 2571 Hag. Md. 21740-2571. Allow 2 to 3 weeks for delivery. NOTE not available for the E or C YET!

MSG LEFT BY: SALLY RIDE
DATE POSTED:

HERE IS THE NEWEST FILE FROM THE LEGION
OF DOOM:
HACKING THE COSMOS PART 1
HERE IS A BRIEF DESCRIIPTION OF THE
MOST COMMONLY USED TRANSACTION CODES:
CAY-CREATE AN ASSEMBLY
DAY-DELETE AN ASSEMBLY
DRE-DENY AND RESTORE ESTABLISHMENT
FLR-FFRAME LAYOUT REPORT
ISH-INQUIRE ABOUT A CIRCUIT(PHONE #)
LO-LIST ORIGINATING LINE EQUIPMENT
MAL-MANUAL ASSIGNMENT LIST
MAY-MODIFY AN ASSEMBLY
MCH-MANUALLY CHANGE HUNT
MDC-MANUALLY DISCONNECT A CIRCUIT
SCA-SERVICE ORDER COMPLETION
SIR-SORTING INQUIRY BY RANGE
SLC-SUBSCRIBER LINE COUNTS FOR
    CUSTOM CALLING FEATURES
US- LIST USOC FILE DATA
WC- WIRE CENTER CHANGE
THOSE WILL BE DISCUSSED IN FURTHER
DETAIL IN PART 2.

PREFIXES, FORMATS AND CODE VALUES:
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
COSMOS PROVIDES A LANGUAGE BY MEAN OF
WHICH A USER CAN COMMUNICATE WITH THE
SYSTEM. THE LANGUAGE INCLUDES VARIOUS
PREFIXES AS WELL AS INPUT FORMATS AND

MSG LEFT BY: SALLY RIDE
DATE POSTED:

INPUT VALUES. PREFIXES ARE ABBREVIATION
WHICH REPRESENT SPECIFIC DATA CATAGORIE
TO THE SYSTEM WHEN INPUT BY THE USER.
AN EXAMPLE OF A PREFIX IS "TN"WHICH
MEANS "TELEPHONE NUMBER". AN INPUT
FORMAT DETERMINES THE NUMBER OF CHARAC-
TERS FOLLOWING A PREFIX AS WELL AS THE
PATTERN IN WHICH THESE MUST BE ENTERED.
FOR EXAMPLE, "TN XXX-XXXX" MEANS THAT
THE PREFIX "TN" MUST BE FOLLOWED BY

```
SEVEN CHARACTERS IN THE FORMAT SHOWN.

INPUT VALUES ARE THE ALLOWABLE DATA
ENTERED FOR EACH PREFIX IN THE CORRECT
INPUT FORMAT. AS MENTIONED IN THE
PREVIOUS PARAGRAPH THE INPUT FORMAT
FOR THE PREFIX "TN" IS "TN XXX-XXXX"
THE FIRST THREE CHARACTERS (XXX) MUST
BE ALPHANUMERIC; THE LAST FOUR (XXXX)
MUST BE NUMERIC. SO, COSMOS WOULD CON-
SIDER AN INPUT OF "TN 935-2481" AS
VALID INPUT. BUT YOU *MUST* USE THE
CORRECT WIRE CENTER FOR THE (XXX) IN
QUESTION. IN HACKING COSMOS PART 2
LEX WILL HAVE A LIST OF THE MOST
COMMONLY USED PREFIXES, FORMATS AND
PREFIX CODE VALUES WHICH ENABLE YOU
TO READ AND UNDERSTAND COSMOS TRANS-
ACTIONS.
SALLY RIDE:::SPACE CADET

MSG LEFT BY: SALLY RIDE
DATE POSTED:

COSNIX IS THE MUTATED VERSION OF COSMOS
AND UNIX BOTH WRITTEN BY BELL LABS.
COSNIX IS THE OPERATING SYSTEM OF THE
COSMOS SYSTEM.
SYSTEM COMMANDS------AS SOME OF YOU
WILL NOTICE, IF YOU READ THE BASICS OF
HACKING II, BY THE KNIGHTS OF SHADOW,
ALOT OF THE COMMANDS USED ON UNIX ARE
ALSO USED ON COSMOS. COMMANDS ARE AS
FOLLOWS::
WHERE---GIVES LOCATION OF THE SYSTEM::
        THIS COMMAND CAN BE VERY USEFUL
        SINCE YOU CAN GO TRASHING AT
        THE LOCATION THAT THE CENTER IS
        AT.
WC%WHERE====COSMOS 5
            STREET ADDRESS
            CITY, STATE  ZIP
WHAT----TELLS WHAT VERSION OF COSNIX
        THE SYSTEM IS RUNNING ON.
WC%WHAT==COSNIX OPERATING SYSTEM9.2.3
         RELEASE DECEMBER 7, 19831.2.2
         ETC.
JUST LIKE ON UNIX, TO SEE WHO ELSE IS
ON THE SYSTEM TYPE: WC%WHO
 COM3    TTOO  GB
 FW6     TTO4  HH, ETC.
COLUMN ONE BEING THE USERNAME, NEXT THE
TT#, AND LAST IS THE WIRE CENTER. SEE
THE CONTINUED CONCLUSION NEXT POSTING.

MSG LEFT BY: SALLY RIDE
DATE POSTED:
```

```
TO SEE WHAT YOU HAVE ACCESS TO TYPE:
WCLLS, OR WC%LS /* TO SEE ALL THE FILES
YOU HAVE ACCESS TO.
USE CAT/FILE-NAME TO SEE SOME MORE INFO
THIS WILL BE EXPLAINED IN MORE DETAIL
IN FUTURE EDITIONS.
DATE==SIMPLY GIVES THE DATE
USING CONTROL C WILL INTERUPT ANYTHING
YOU ARE EXECUTING AT THE TIME. YOU MAY
HAVE TO ENTER IT MORE THAN ONCE.

THAT IS ALL FOR PART 1, IT SHOULD GIVE
YOU A BASIC UNDERSTANDING OF COSMOS.
PART 2 WILL EXPLAIN AND SHOW YOU HOW TO
HOOK UP PHONE #'S AND SEE WHAT EQUIPMNT
IS ATTACHED, ALSO, WHAT THE ABBREVIA-
TIONS ARE SO YOU CAN UNDERSTAND IT ALL.

ACKNOWLEDGEMENTS: THE WARLOCK
                  BIOC AGENT 003
                  TUC-TUCBBS
WRITTEN BY: LEX LUTHOR

UPLOADED HERE BY:
     SALLY RIDE:::SPACE CADET
POST SOME DAMN MESSAGES YOU SLUGS!

APPLE CLONES

64K
CPM
NUMERIC KEYPAD
FUNCTION KEYS
AUTO REPEAT
100% APPLE COMPATABLE

$650

IBM CLONES
128K RAM
ROM CHIPS NOT INCLUDED
AVAILABLE IN PC OR XT VERSIONS

$650

CALL FOR DETAILS- 503-592-4461

NUFF SAID-
BOOTLEG

*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
       LOCKSMITH PARAMETERS LIST
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

     DONATED BY: SHERLOCK APPLE

ZORK (OLD VERSIONS)
     T0-T22: 1E=0B
     T3:    4C=1B (PATCH NC30 FOR VERSI
ON 4.0)
            4C=1B 57=00
E9=02  (USES NIBBLE COUNT SEE TECH NOTE
S) (VERSION 4.1 ONLY)
ZORK I AND ZORK II (NEW VERSIONS)
     T0-T22: 46=96 40=14

WARP FACTOR **     SAME AS TORPEDO FIRE
.FF4
WORD HANDLER
     T0: 46=96 54=12
-T22
     T1-TC: 44=FF 45=DF 46=DE (8 ERRORS
 O.K.)
.FF5
WORD HANDLER II
     T0: 46=96 54=12 53=00
     T11-T22
     T1-TC: 44=FF 45=DF 46=DE
     (NOTE-IF
AN 8 ERROR OCCURS RECOPY TRACK IT HAPPE
NED ON UNTIL GOOD.)

VISICALC (DOS 3.3 VERSION)
     T0-T15 NORMAL (T1 ERR IS OK)
VISICALC (APPLE ///)
     T0-T22 SYNC
VISIDEX (CHANGE AS OF 11-18-81)
     T0-T22: 40=04 16=08 41=FF 19=00 58
=0B 59=FF 81=AA 82=EB
83=FD 21=02
             46=96 54=12
VISIFILE     SAME AS DESK TOP PLAN II E
XCEPT PARM C0=FD SHOULD BE
C0=EC
VISISCHEDULE
     T0-T22: 40=04 16=08 41=FF 19=00 58
=0B 59=FF 81=AA 82=EB
83=EC 21=02 46=96 54=12
.FF3
VISITERM
     T0-T22 NORMAL
     T6: 40=08 16=08 41=FF 19=00 58=0B
59=FF 81=AA 82=EB 83=FC
.FF3
VISITREND/VISIPLOT
     T0-T22 NORMAL
     T7: 40=08 16=08 41=FF 19=00 81=DE
82=AA 58=0B 59=FF

U-BOAT COMMAND  **
     T0-T22: 4E=00 51=00 52=00 40=02 1E
=30 1B=19 1D=18 44=00
45=00 46=EB 47=AF
             48=FB 49=EB
.FF2
ULTIMA
```

```
     T0-T22: 1E=0B
ULYSIS **
     T0-T22 NORM
     T3: 4C=1B APPLY PATCH NC30 (VERSIO
N4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)

TAX PREPARER
     T0-T22: 46=96 54=12 4C=19
.FF4
THRESHOLD
     T0-T22 NORMAL
     T1-T23 BY 22: 4C=1B (PATCH NC30 FO
R VERSION 4.0)
                   4C=1B 57=00 E9=02  (
USES NIBBLE COUNT SEE TECH
NOTES) (VERSION 4.1 ONLY)
.FF2
TINY TROL
     T0-T22 NORMAL     T3.5-T5 BY 1.5
.FF2
TORPEDO FIRE
     T0 NORMAL     T1-T22: 4F=0B
.FF3
TWERPS **
     SAME AS GORGON
     PLUS T1C: 4C=1B 57=00 E9=02 D2=00
TWERPS  **
     T0: 18=20 19=00 46=96 4D=00 4E=00
52=00 53=00 54=12 57=00
40=20
     T1.5-TE.5 BY 1 SYNC: 72=00 73=00 7
7=00 78=00 79=12 7C=00
44=DD 45=AD 46=DA

SABATOGE **
     T0-T22 NORM
     T3: 4C=1B  APPLY PATCH NC30 (VERSI
ON 4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
SARGON II **
     T0-T1A NORM: 19=00 54=12 47=FF 4C=
18 48=FF 50=00 51=00 52=00
53=00
SCREENWRITER II **
     T0-T2: 4D=00
SHATTERED ALLIANCE
     T0-T22: 25=19
SHATTERED ALLIANCE (NEW)
     T0: 4C=18 47=FF 53=0B 54=12
     T1-T22: 44=D4 46=B7
.FF2
SINGA SHAPE MANAGER **
     T0-T22 SYNC
```

```
SNAKEBITE ** SAME AS GORGON
SNEAKERS
     T0: 18=20 19=00 46=96 4D=00 4E=00
52=00 53=00 54=12 57=00
40=20
     T1.5-TD.5 BY 1 SYNC: 72=00 73=00 7
7=00 78=00 79=12 7C=00
40=20 19=00 44=DD 45=AD 46=DA
.FF5
SNOGGLE **
     T0-T9 NORM
        OR
     T0-TF NORM     T10.5-T11.5 SYNC
.FF4
SOFTPORN ADVENTURE
     T0-T22 NORMAL (ALL VERSIONS)
     T3: 4C=1B APPLY PATCH NC30 (VERSIO
N 4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
.FF2
SOUTHERN COMMAND **
     T0-T22: 25=19 6B=00 34=D5 35=AB
.FF3
SPACE EGGS
     T0 NORM     T2-6 NORM     T11-13 N
ORM
     T14-1A: 44=DD
SPACE QUARKS
     T0: 18=50 19=00 40=20 46=96 4D=00
4E=00 52=00 53=00 54=12
57=00
     T1-T2: 44=AB 45=D4 46=AB
     T3.5-T5.5 BY 1     T7
     T9: 44=FE 45=DD 46=AF
     TA.5-B.5 BY 1: 44=AA 45=DE 46=BB
     TD-15 BY 1
SPACE WARRIOR
     T0: 18=50 19=00 40=20 46=96 40=20
4E=00 52=00 53=00 54=12
57=00
     T2.5-T3.5: 44=DF 45=AD 46=DE
     T5-T8 BY 3     T6.5     TA-T10 BY
3
STAR BLASTER  **
     T0 NORM
     T7-T20.5 BY 1.5 SYNC: 72=00 73=00
77=00 78=00 79=12 7C=00
40=20 19=00 44=DF 45=AD 46=DE
STAR CRUISER **
     T0-T3 BY 3 SYNC     T5-TB BY 1 SYN
C     T11-T12 BY 1 SYNC
     T4 SYNC: 44=AA 45=DD 46=BB
STAR MINES **
     T0 NORM
     T1-T2 NORM: 46=AD
     T4-TA NORM
```

```
STAR RATERS **
     T0-T5 NORM (TRACK 5 ERROR MAY OCCU
R)
STAR THIEF
     T0-T13 NORMAL (TRACK E-13 ERRORS M
AY OCCUR) (ALL VERSIONS)
     T22: 4C=1B APPLY PATCH NC30 (VERSI
ON 4.0 ONLY)
          4C=1B 57=00 E9=02  (USES NIBB
LE COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
.FF2
SUPER APPLE BASIC **
     T0-T22 NORM     T3 NORM-EXTENDED R
ETRY
.FF3
SUPERSCRIBE II
     T0-T22 NORM
     T3 NORM: 45=00 50=00
SUPERSCRIBE II **     SAME AS PEGASUS I
I
.FF2

RASTER BLASTER (FOR OLD RASTER BLASTER
ONLY)
     T0 NORMAL
     T5-T11 BY 4 SYNC: 44=AD 45=DE 53=0
0
     T6-T12  BY 4 SYNC     T7.5-TF.5 BY
 4 SYNC     T1.5-T3.5 BY 2
SYNC
.FF4
RASTER BLASTER (NEW VERSIONS)
     T0: 46=96 54=12
     T5-T11 BY 4 SYNC: 44=AD 45=DE 46=0
0 72=00 73=00 75=00 78=00
79=12
     T6-T12 BY 4 SYNC     T7.5-TF.5 BY
4 SYNC     T1.5-T3.5 BY 2
SYNC
.FF3
RETROBALL  **
     T0, T4-T6, T
----------------------------------------

YEP-WEVE BEEN A LITTLE LATE WITH THIS
ISSUE DUE TO MOVING.
STILL HAVE THE SAME PHONE NUMBERS,
BUT OUR CURRENT ADDRESS TIS-

     THE BOOTLEGGER/HACKER MAGAZINE

         1080 HAYS CUT-OFF ROAD
         CAVE JUCTION,OR.97523
2
```

```
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

[[ PRESS SPACEBAR TO QUIT ]]

        NIBBLES AWAY  PARAMETERS

COMPANY NAME:
                    AUTO-LOAD FILE
PROGRAM NAME        COPY TRACKS    PARA
METERS TO CHANGE        TO USE
----------------    -----------    ----
----------------   ----------------
A P P L E   C O M P U T E R
SUPER PILOT ------- 0-0............ADDR
=D5 AA 96
                    2-22
                        SECTMOD [F=16,C
=OFF,T=0,S=0A]
                          CHANGE ADDRES
S 79 FROM 43 TO EA
                          CHANGE ADDRES
S 7A FROM 41 TO EA
                          CHANGE ADDRES
S 7B FROM C6 TO EA

A U T O M A T E D   S I M U L A T I O N
 S:
TEMPLE OF APSHAI -- 0-22...........ADDR
=D5 AA B5

A V A N T E - G A R D E
HI-RES SECRETS ---- 0-22...........ADDR
=D5 AA 96

B R O D E R B U N D   S O F T W A R E:
WARLORDS ---------- 0-F............ADDR
=D5 AA B5

C E N T R A L   P O I N T   S O F T W A
 R E:
COPY ][ PLUS ------ 0-2............NORM
AL
                                   DEL
BYTE =20
D A T A   M O S T:
SPACE KADET ------- 0-22...........ADDR
=D5 AA 96
MARS CARS                          OVER
IDE STANDARDIZER
CRAZY MAZEY
TAX BEATER -------- 0-22...........ADDR
=D5 AA 96
REAP                    SECTMOD [F=16,C
=OFF,T=0,S=03]
                          CHANGE ADDRES
S 42 FROM 38 TO 18
MONEY MUNCHER ----- 0-22...........ADDR
=D5 AA 96
```

```
E D U W A R E:
THE PRISONER ------ 0-22...........SYNC
ALGEBRA I --------- 0-22...........ADDR
=D5 AA B5
EMPIRE 1 WORLD ---- 0-22...........ADDR
=D5 AA 96
BUILDERS            3-3............NIBB
LE COUNT
PRISONER ][ ------- 0-22...........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=ON,T=1F,S=0E]
                         CHANGE ADDRESS
 D5 FROM AD TO 2F
                         CHANGE ADDRESS
 D6 FROM 99 TO AF
                         CHANGE ADDRESS
 D7 FROM F0 TO 32
I N F O C O M:
STARCROSS --------- 0-22...........ADDR
=D5 AA 96

I N S O F T:
ELECTRIC DUET ----- 0-22...........ADDR
=D5 AA 96
                                   INS=
 DE AA EB
                                   OVER
IDE STANDARDIZER
                                   FIX
AMNT=04
I N T ' L   S O F T W A R E   M K T G
MATH MAGIC -------- 0-22...........NORM
AL

I D S:
PRISM PRINT ------- 0-21...........ADDR
=D5 AA 96
                                   OVER
IDE STANDARDIZER
                        SECTMOD [F=16,C
=ON,T=21,S=00]
                          CHANGE ADDRES
S 27 FROM FB TO 22

L E A R N I N G   C O M P A N Y
BUMBLE GAMES ------ 0-22...........ADDR
=D5 AA 96
BUMBLE PLOT         NOTE: WRITE PROTECT
 BEFORE BOOTING!
ROCKY'S BOOTS
JUGGLER'S RAINBOW

M I C R O L A B
JIGSAW ------------ 0-0............NORM
AL
                    A-17...........NORM
```

```
AL
                    1-9...........ADDR
=D3 96 F2
M U S E:
BEST OF MUSE ------ 0-22..........SYNC
THREE MILE ISLAND
GLOBAL WAR

M I C R O S O F T:
OLYMPIC DECATHALON  0-22..........ADDR
=D5 AA B5

O N L I N E   S Y S T E M S:
GENERAL MANAGER --- 0-22..........ADDR
=D5 AA 96
V1.5                    SECTMOD [F=16,C
=ON,T=1F,S=0E]
                          CHANGE ADDRES
S C1 FROM -- TO 4B
                          CHANGE ADDRES
S C2 FROM -- TO E0
                          CHANGE ADDRES
S C3 FROM -- TO 49
                        SECTMOD [F=16,C
=ON,T=21,S=01]
                          CHANGE ADDRES
S 2E FROM -- TO 60
SABOTAGE ---------- 0-22..........NORM
AL
ALIEN RAIN
SNOGGLE ----------- 0-22..........ADDR
=D5 AA B5

TIME ZONE V1.1 ---- 0-22..........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=ON,T=03,S=0B]
                         CHANGE ADDRESS
 F0 FROM 20 TO EA
                         CHANGE ADDRESS
 F1 FROM 00 TO EA
                         CHANGE ADDRESS
 F2 FROM 17 TO EA

P E N G U I N   S O F T W A R E:
PIE MAN ----------- 0-22..........ADDR
=D5 AA 96

P H O E N I X   S O F T W A R E:
ZOOM GRAPHICS ----- 0-22 BY 2.....ADDR
=D5 AA 96
2ND EDITION                       INS=
DD AA ED B5
                    1-21 BY 2.....ADDR
=D4 AA 96
                    N O T E: WRITE PROT
ECT BEFORE BOOTING!!
```

```
ADVENTURE IN TIME - 0-C...........NORMA
L
BIRTH OF THE ------ 0-9...........NORMA
L
PHOENIX

P I C A D I L L Y   S O F T W A R E:
FALCONS ----------- 0-0............ADDR
=D5 AA B5
                    1.5-4.5X1.5....ADDR
 DF AD DE
                    5.5-5.5X1
                    7-AX1
                    B.5-E.5X1.5
                    10-12X1
                    13.5-14.5X1
                    16-19X1.5
                    1A-1B.5X1.5

S E N S I B L E   S O F T W A R E:
IMAGE PRINTER ----- 0-2............ADDR
=D5 AA 96
                    3-7............ADDR
=F7 AA 96
                    9-22
                     SECTMOD [F=16,C=OF
F,T=0,S=03]
                       CHANGE ADDRESS 4
2 FROM 38 TO 18
                     SECTMOD [F=16,C=OF
F,T=2,S=03]
                       CHANGE ADDRESS 2
A FROM 2C TO 4C
                       CHANGE ADDRESS 2
B FROM 06 TO 5D
                       CHANGE ADDRESS 2
C FROM B7 TO B4
SUPER DISK COPY --- 0-22...........ADDR
=D5 AA 96
(VERSION 3.7)                       ERR
ORS OK
THE BUG ----------- 0-0............NORM
AL
                    15-15..........GAP
BYTE 2=FF
                                   GAP
SIZE=10
                    16.5-16.5

S E R I U S   S O F T W A R E:
KABUL SPY --------- 0-21...........ADDR
=D5 AA 96
(BOTH SIDES)         SECTMOD [F=16,C=OF
F,T=0,S=0
                       CHANGE ADDRESS 4
9 FROM -- TO EA
                       CHANGE ADDRESS 4
A FROM -- TO EA
```

```
                       CHANGE ADDRESS 4
B FROM -- TO EA
DARK FOREST ------- 0-22...........ADDR
=D5 AA B5
                                   OVER
IDE GLITCH DETECT

S I L I C O N   V A L L E Y   S O F T W
 A R E:
WORD HANDLER ][ --- 0-0C...........ADDR
=FF DF DE
                    11-22..........ADDR
=D5 AA 96
S O F T A P E:
DRAW POKER -------- 0-22...........ADDR
=D5 AA B5

S O F T W A R E   P U B L I S H I N G
 C O R P.:
PFS/PFS REPORT ---- 0-13...........ADDR
=D5 AA 96
(REVISED)                          OVER
IDE STANDARDIZER
                                   GAP
BYTE 1=C0, GAP BYTE 2=D0
                                   FILT
ER=C0-C8 (NO INVERSE)
                    N O T E: WRITE PROT
ECT BEFORE BOOTING!!
PFS GRAPH --------- 0-22...........ADDR
=D5 AA 96
                                   OVER
IDE STANDARDIZER
                                   GAP
BYTE 1=C0, GAP BYTE 2=D0
                                   FILT
ER=C0-C8 (NO INVERSE)

S P E C I A L   D E L I V E R Y   S O F
 T W A R E:
UTOPIA GRAPHICS --- 0-22...........ADDR
=D5 AA 96
SYSTEM                             TURN
 ON 3.3 FILTER
                       SECTMOD [F=16,C=
ON,T=0,S=0]
                         CHANGE ADDRESS
 42 FROM 38 TO 18
GALACTIC WARS ----- 0-22...........ADDR
=D5 AA 96
BRIDGE TUTOR

S T O N E W A R E:
D B MASTER -------- 0-5............ADDR
=D5 AA 96, SYNC
UTILITY PAC #1      6.5-22.5.......SYNC

S T R A T E G I C   S I M U L A T I O N
```

```
 S:
BATTLE OF SHILOH -- 0-22...........ADDR
=D4 AA B7
WARP FACTOR

S Y T O N I C   S O F T W A R E:
INTERLUDE ----------0-22...........ADDR
=D5 AA B5

X P S:
APPLE CILLIN ------ 0-0............ADDR
=D5 AA 96
                    1-22...........ADDR
=D5 AA B5
                    11-11..........ADDR
=D5 AA 96

                PARAMETERS:  OCTOBER 19
82

COMPANY NAME:
PROGRAM NAME        COPY TRACKS    PARA
METERS TO CHANGE
----------------    -----------    ----
----------------
A D V E N T U R E   I N T E R N A T I O
 N A L:
ELIMINATOR -------- 0-21...........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=OFF,T=03,S=0D]
                          CHANGE ADDRES
S 2E FROM 20 TO EA
                          CHANGE ADDRES
S 2F FROM 30 TO EA
                          CHANGE ADDRES
S 30 FROM 72 TO EA

A P P L E   C O M P U T E R:
VISICALC /// ------ 0-22...........SYNC
APPLE WRITER /// -- 0-22...........SYNC
APPLE LOGO -------- 0-22...........ADDR
 D5 AA 96
                    1-1............ADDR
 AA D6 EE
                                  NIBBL
E COUNT=Y
                                     FI
ND MAX=03
                                   SHIF
T N+ = 08
                                   SHIF
T N- = 00

APPLE WRITER ][ --- 0-3............ADDR
 D5 AA DA (OR D5 AA DB)
                    4-22...........ADDR
 D5 AA 96
```

```
A V A N T E - G A R D E   C R E A T I O
 N S
ZERO GRAVITY PINBALL 0-22...........ADD
R=D5 AA B5

B P I:
   (REVISED)
ACCOUNTING -------- 0-22...........ADDR
=D5 AA 96
  SYSTEM                    FIX AMNT=04
,  GAPBYTE1=C8
                            GLOBAL MOD
BYTE D972 FROM 03 TO 00
                    11-11..........INS=
AD FB E6 FF E6
                                   SYNC
 SIZ=0A

B R O D E R B U N D   S O F T W A R E:
APPLE PANIC ------- 0-D
GENETIC DRIFT ----- 0-0............ADDR
=D5 AA B5
                    1-3............ADDR
=BB D5 BB
                    4.5-6 BY 1.5
                    7.5-B.5
                    D-D............ADDR
=D4 D5 BB
                    E.5-12.5.......ADDR
=AD B5 DE

SPACE QUARKS ------ 0-0............ADDR
=D5 AA B5
                    1-2............ADDR
=FF DF DE, DATA MAX=25
                    3.5-5.5
                    7-9 BY 2
                    A.5-B.5
                    D-15

SPACE WARRIOR ----- 0-0............ADDR
=D5 AA B5, DATA MAX=30
                    2.5-3.5........ADDR
=DF AD DE
                    5-8 BY 3
                    6.5-6.5
                    A-10 BY 3

B U D G C O:
RASTER BLASTER ---- 0-0............ADDR
=D5 AA 96, SYNC
                                   DATA
 MIN=18, DATA MAX=40
                    5-11 BY 4......ADDR
=AD DE, DATA MIN=13, SYNC
                    6-12 BY 4......SYNC
                    7.5-F.5 BY 4...SYNC
                    1.5-3.5 BY 2...SYNC

C A V A L I E R   C O M P U T E R:
MICROWAVE --------- 0-22...........ADDR
=D5 AA 96
                      SECTMOD [F=16,C=O
N,T=02,S=01]
                         CHANGE ADDRESS
 DA FROM A9 TO AD
                         CHANGE ADDRESS
 DB FROM 60 TO 03
                         CHANGE ADDRESS
 DC FROM 8D TO 81
                         CHANGE ADDRESS
 DD FROM 7E TO 60

C O N T I N E N T A L   S O F T W A R E
:
GUARDIAN ---------- 0-1............ADDR
=D5 AA B5
                    2-11...........ADDR
=D6 AA B5
                                   INS=
DF AA EB F7, SYNC SIZ=0A
D A T A   M O S T:
COUNTY FAIR ------- 0-22...........ADDR
=D5 AA B5
SNACK ATTACK         SECTMOD [F=13,C=OF
F,S=03,T=00]
                       CHANGE ADDRESS 6
3 FROM 38 TO 18
SNACK ATTACK ------ 0-22...........ADDR
=D5 AA B5
(REVISED)            SECTMOD [F=13,C=OF
F,S=01,T=00]
                       CHANGE ADDRESS 3
9 FROM 38 TO 18

SWASHBUCKLER ------ 0-22...........ADDR
=D5 AA 96
CASINO 21            SECTMOD [F=16,C=OF
F,S=03,T=00]
                       CHANGE ADDRESS 4
2 FROM 38 TO 18

D A T A   S O F T:
DUNG BEETLES ------ 0-0............ADDR
=D5 AA B5
                    1-1............ADDR
=F5 F6 F7
                    4-22
                     SECTMOD [F=13,C=ON
,T=00,S=01]
                         CHANGE ADDRESS
 6D FROM 01 TO 7B
                         CHANGE ADDRESS
 6E FROM 61 TO 69
```

```
G E B E L L I   S O F T W A R E:
FIREBIRD ---------- 0-0...........ADDR
=DD AD DA, SYNC
                    1.5-B.5.......SYNC

H O W A R D S O F T:
TAX PREPARER ------ 0-22..........ADDR
=D5 AA 96

I N F O C O M:
DEADLINE ---------- 0-22..........ADDR
=D5 AA 96

I N N O V A T I V E   D E S I G N   S O
 F T W A R E:
POOL 1.5 ---------- 0-15..........ADDR
=D5 AA B5
                    1E-21
                      SECTMOD[F=13,C=OF
F,T=0B,S=07]
                         CHANGE ADDRESS
 6A FROM 8D TO 60
```

STEP BY STEP GUIDE TO BACKING-UP DISKS
WITH
NIBBLES AWAY][

THERE ARE THREE BASIC STEPS TO BACKUP A DISKETTE:

1. LOCATE THE TRACKS WHICH CONTAIN DATA.
2. FIND THE ADDRESS MARKER FOR THE SECTORS THERE.
3. FIGURE OUT ANY ADDITIONAL PROTECTION.

(HINT: #3 IS THE HARD ONE!)

FOR MOST OF THE PROCEDURES BELOW, A BASIC WORKING KNOWLEDGE OF THE TRACK/BIT EDITOR (TBE) IS REQUIRED. FOR THOSE WHO ARE NOT FAMILIAR WITH THE TBE, AN OVERALL DESCRIPTION AND SOME EXAMPLES ARE GIVEN BELOW. THE EXAMPLES ARE EASIER TO UNDERSTAND IF THEY ARE PERFORMED AS YOU READ THIS, SO YOU MAY WANT TO BOOT UP NIBBLES AWAY][AND TRY THEM OUT TO GET A BETTER UNDERSTANDING OF WHAT IS GOING ON.

ENTER THE TBE BY SELECTING OPTION 'T' FROM THE MAIN MENU. A LARGE SECTION OF NUMBERS WILL APPEAR ON THE SCREEN, WITH TWO DASHED LINES AT THE TOP. THE INFORMATION IN BETWEEN THESE LINES IS THE STATUS INFORMATION AND INFORMS YOU OF SUCH THINGS AS CURSOR POSITION, TRACK NUMBER, AND IS ALSO THE LOCATION WHERE VARIOUS PROMPTS APPEAR FOR CERTAIN FUNCTIONS. THE NUMBERS AT THE BOTTOM ARE SEPARATED INTO TWO SECTIONS. ON THE LEFT ARE THE STARTING MEMORY ADDRESS'S FOR EACH LINE TO THE RIGHT. MOVE THE CURSOR AROUND USING I,J,K OR M, AND WATCH THE ADDR INDICATOR IN THE STATUS LINE. IT WILL TELL YOU EXACTLY WHAT MEMORY ADDRESS THE VALUE UNDER THE CURSOR REPRESENTS. THE ARROW KEYS CHANGE THE AREA OF MEMORY WHICH YOU CAN SEE. THEY SHIFT YOUR VIEW 256 BYTES FORWARD OR BACKWARD AT A TIME. THE ONLY REALLY IMPORTANT THING TO KNOW FOR THIS DISCUSSION IS HOW TO USE THE ARROW KEYS TO MOVE THE VIEWING 'WINDOW' AROUND IN MEMORY.

THE ';' (UNSHIFTED '+') AND THE '-' KEYS INCREMENT AND DECREMENT THE

TRACK NUMBER IN THE STATUS LINE. PRESSING 'R' WILL CAUSE DRIVE ONE TO READ THE DATA FROM THE TRACK INDICATED IN THE STATUS LINE INTO MEMORY. THE BYTES ON THE SCREEN WILL CHANGE, SINCE DIFFERENT DATA HAS BEEN READ IN. PRESSING THE 'R' KEY MULTIPLE TIMES WILL RESULT IN DIFFERENT DATA BEING DISPLAYED. THIS IS BECAUSE NIBBLES AWAY][STARTS READING AT WHATEVER POINT HAPPENS TO BE UNDER THE HEAD WHEN THE DRIVE IS TURNED ON, WHICH IS RANDOM, HENCE THE CHANGE IN THE DISPLAYED DATA (THE DATA IS NOT ACTUALLY DIFFERENT, IT IS JUST NOT LOADED AT THE SAME MEMORY LOCATION AS IT WAS PREVIOUSLY).

STEP 1:
TO DO THIS WE MUST LOCATE ALL OF THE TRACKS ON THE DISK WHICH CONTAIN DATA. TO DO THIS WE SHOULD HAVE THE TRACK POINTER SET TO TRACK 00. PRESSING 'R' WILL READ IN THE TRACK AND SHOW IT ON THE SCREEN. THE ARROW KEYS SHOULD BE USED TO MOVE THE VIEWING 'WINDOW' TO START AT $2000. NOW WE WILL MOVE FORWARD AND TRY TO DETERMINE IF THIS TRACK CONTAINS VALID DATA. ACTUALLY, TRACK 00 MUST CONTAIN SOME DATA IN ORDER FOR THE DISK TO BOOT, BUT WE WILL BE USING THIS PROCEDURE ON OTHER TRACKS WHICH DO NOT NECESSARILY CONTAIN DATA.

THE MAIN THING WHICH WILL IDENTIFY A TRACK AS CONTAINING DATA IS THE PRESENCE OF GAPS. GAPS ARE SECTIONS OF THE SAME BYTE REPEATED SEVERAL TIMES. NORMALLY THEY ARE MADE UP OF $FF'S AND ARE 6-20 IN LENGTH. TO SEE WHAT THESE LOOK LIKE, INSERT YOUR SYSTEM MASTER DISK AND READ IN TRACK 00 AS DESCRIBED ABOVE. MOVING THROUGH THE BUFFER WITH THE ARROW KEYS WILL REVEAL A LARGE VARIETY OF VALUES. SPACED OUT AMONG THESE SHOULD BE SECTIONS OF FF'S WHICH CONTAIN ABOUT 6-20 IN A ROW, DEPENDING ON THE EXACT DISK. NORMALLY DOS 3.2 DISKS HAVE LARGER GAPS THAN DOS 3.3 DISKS. THERE SHOULD BE MANY OCCURANCES OF THE GAPS, SPACED OUT SO THAT YOU SEE ONE ABOUT EVERY OTHER TIME THAT YOU USE THE ARROW KEYS TO MOVE FORWARD OR BACKWARD.

NOTE:YOU MAY SEE A SECOND, SMALLER (2-5 $FF'S), GAP FOLLOWING A LARGE GAP, WITH A SMALL SECTION OF DATA IN BETWEEN. THIS IS CALLED THE SECONDARY GAP. WHEN REFERING TO A GAP HERE, WE WILL ALWAYS BE TALKING ABOUT THE PRIMARY GAP, NOT THE SECONDARY ONE.

NOW TRY LOOKING AT OTHER TRACKS ON THE DISK. FIRST LOOK ONLY AT THE FULL TRACKS (NO .5 ON THE END). ALL OF THEM WILL BE SIMILAR TO TRACK 00 IN THE APPEARANCE OF THE GAPS. YOU MAY WANT TO TRY THIS SEVERAL TIMES TO BECOME COMFORTABLE WITH LOCATING GAPS ON A GIVEN TRACK.

NOW READ IN A HALF TRACK (.5 ON THE END). SCAN MEMORY TO LACATE SOME OF THE GAPS. SINCE SYSTEM MASTER DISKS DO NOT USE HALF-TRACKS, THE DATA WHICH WE SEE HERE IS REALLY 'CROSS-TALK'. IN OTHER WORDS, DATA WAS WRITTEN ON THE FULL TRACK, BUT THE MAGNETIC PATTERN SPREAD OUT A BIT, AND SO WE SEE SOME DATA HERE. THE TELL-TALE SIGN OF THIS PHENOMENA IS THAT THE GAPS WILL NOT BE ALL THE SAME. THAT IS, THEY MAY HAVE ONE OR MORE VALUES IN THEM WHICH ARE NOT CONSISTENT. THIS TELLS US THAT THERE IS SOME DATA ON THIS TRACK, BUT THAT IT IS NOT VALID DATA. TAKE A LOOK AT SOME OTHER HALF-TRACKS SO THAT YOU CAN TELL IF YOU ARE LOOKING AT A FULL TRACK OR A HALF TRACK BY EXAMINING THE GAPS.

THE NEXT ITEM WHICH YOU NEED TO BE ABLE TO IDENTIFY IS A BLANK TRACK. TO DO THIS, INSERT A BLANK (NON-INITIALIZED) DISK INTO DRIVE ONE. READ ANY TRACK ON THIS DISK AND SCAN THROUGH THE MEMORY ADDRESSES. THERE WILL BE NO GAPS FOUND, AND MANY OF THE BYTES SEEN ON A TRACK LIKE THIS WILL END IN 0 (I.E. $A0,$B0,$E0), WHICH ARE NOT LEGAL DISK BYTES. THIS MEANS THAT THE CONTROLLER CAN FIND NO VALID DATA ON THE TRACK. SOME DISKS HAVE PORTIONS OF TRACKS WHICH ARE NOT USED, SO YOU SHOULD ALWAYS BE SURE TO EXAMINE AT LEAST 24 SCREENFULLS OF INFORMATION TO MAKE SURE THAT THERE IS NO DATA AT ANY POINT ON THE TRACK.

OUR NEXT TOOL FOR FINDING DATA IS THE FACT THAT VALID DATA MUST BE AT LEAST 1 TRACK APART. IN OTHER WORDS, IF YOU LOCATE DATA ON TRACK 3.5, THEN TRACK 4 CANNOT HAVE DATA AND THE NEXT PLACE WHERE DATA CAN BE IS TRACK 4.5. THIS IS VERY HELPFUL FOR FINDING TRACKS WITH DATA.

NOTE: IF YOU LOCATE DATA ON A GIVEN TRACK, IT IS A GOOD IDEA TO LOOK AT THE TRACKS ONE HALF TRACK TO EITHER SIDE, TO MAKE SURE THAT THEY LOOK LESS VALID THAN THE ONE THAT YOU HAVE SELECTED AS THE REAL ONE.

WELL, NOW THAT WE KNOW HOW TO LOACATE DATA ON A TRACK, WE CAN BEGIN AT TRACK 0 AND STEP TOWARDS TRACK 22, CHECKING EACH TRACK TO SEE IF IT APPEARS TO HAVE DATA ON IT. MOST DISKS HAVE A PATTERN TO THE POSITION OF THE DATA, AND IF YOU CAN FIGURE IT OUT, YOU MAY BE ABLE TO JUST CHECK A FEW TRACKS TO MAKE SURE, AND THEN GO ON TO STEP 2. OTHERWISE THE DATA MUST BE LOCATED ONE TRACK AT A TIME.

MOST DISKS USE THE STANDARD TRACKS (1,2,3,...,22), BUT THERE ARE SOME WHICH USE HALF-TRACKS AND SOME WHICH USE ALL THE WAY OUT TO TRACK 23 (WHICH, BY THE WAY CANNOT BE READ ON ALL DRIVES SINCE NO DRIVES WERE EVER DESIGNED TO GO OUT THAT FAR).

WHEN ALL TRACKS WHICH CONTAIN SOME TYPE OF DATA ARE LOCATED, WE CAN MOVE ON TO STEP 2.

STEP 2:

NOW WE MUST TELL NIBBLES AWAY][HOW TO READ THE INFORMATION ON THE TRACKS WHICH WE HAVE FOUND TO CONTAIN VALID DATA. THIS IS DONE BY GOING BACK TO EACH OF THESE TRACKS WITH THE TBE AND FINDING THE ADDRESS MARK FOR EACH ONE. THE ADDRESS MARK WILL BE THE FIRST 3 BYTES FOLLOWING THE GAP. TO SEE THIS IN OPERATION, TAKE A LOOK AT A TRACK FROM YOUR SYSTEM MASTER DISK. AFTER EACH GAP YOU WILL SEE EITHER 'D5 AA 96' FOR A DOS 3.3 MASTER DISK, OR 'D5 AA B5' FOR A DOS 3.2 DISK. THESE VALUES SHOULD BE NOTED DOWN ALONGSIDE OF EACH TRACK NUMBER WHICH CONTAINS DATA. MANY TIMES THERE WILL BE ONLY ONE, OR MAYBE 2 PATTERNS FOR ALL TRACKS.

AFTER THIS, WE ARE READY TO BACK-UP THESE TRACKS. THIS IS DONE BY EXITING THE TBE (USE 'Q') AND THEN SELECTING 'M' FOR THE MODIFIERS MENU. THEN SELECT 'B' FOR BACKUP MODIFIER. WHEN ASKED 'USE ADDRESS MARK' ANSWER 'Y' AND THEN TYPE IN THE ADDRESS MARK WHICH YOU NOTED DOWN FOR THE RANGE OF TRACKS TO BE BACKED-UP. SIMPLY PRESS RETURN TO THE REST OF THE QUESTIONS AND THEN RETURN TO THE MAIN MENU. SELECT 'N' TO ENTER NIBBLES AWAY][, AND ANSWER 'Y' TO THE QUESTION 'CHANGE DEFAULT OPTIONS'. USE THE <RETURN> KEY TO MOVE TO THE 'START TRACK' PROMPT, AND THEN ENTER THE FIRST TRACK TO BE BACKED-UP. PRESS RETURN AND THEN TYPE IN THE LAST TRACK TO BE BACKED-UP WITH THE CURRENT ADDRESS MARKER SETTING. IF THE TRACKS IN THE SPECIFIED RANGE ARE NOT SPACED AT 1 TRACK INTERVALS, ENTER THE INTERVAL AT THE 'TRACK INCREMENT' PROMPT. PRESS RETURN FOR THE FOLLOWING QUESTIONS AND BEGIN THE BACKUP AFTER INSERTING THE DISKS WHEN PROMPTED. WHEN YOU RETURN TO THE MAIN MENU, REPEAT THE ABOVE PROCEDURE FOR EACH RANGE OF TRACKS WHICH CONTAINS A DIFFERENT ADDRESS MARKER.

NOW COMES THE MOMENT OF TRUTH! TRY TO BOOT UP THE BACKED-UP DISK (IF THE ORIGINAL HAD A WRITE-PROTECT TAB, THE BACK-UP SHOULD TOO!). IF THE BACKUP BOOTS, THEN ALL WENT SUCCESFULLY.

STEP 3:

IF THE BACK-UP DID NOT WORK PROPERLY THEN THERE ARE A FEW THINGS TO LOOK FOR.

1....DID ALL OF THE TRACKS WHICH SHOULD HAVE BACKED-UP DO SO? THIS CAN BE SEEN WHILE THE BACK-UP TAKES PLACE AS A 'Y' OR AN 'N' UNDER THAT TRACKS STATUS LOCATION. IF SOME DID NOT, THEN THE ADDRESS MARKER WAS PROBABLY NOT DETERMINED PROPERLY. IF THIS IS THE CASE, THEN GO BACK TO THE TBE AND TRY THOSE TRACKS AGAIN.

2....IF EVERYTHING SEEMED TO GO WELL, BUT THE BACKUP REFUSES TO WORK (YOU MAY WANT TO TRY THE PROCEDURE AGAIN, MAYBE WITH THE SOURCE AND DESTINATION DRIVES REVERSED, TO MAKE SURE IT WAS NOT A POWER GLITCH OR OTHER SUCH

OCCURANCE WHICH MESSED THINGS UP) THE NEXT STEP IS TO TRY THE PROCEDURE WITH THE 'SYNCHRONIZED COPY' OPTION SELECTED. DISKS WHICH USE THIS METHOD OFTEN MAKE VIOLENT HEAD MOVEMENTS DURING THEIR BOOT PROCEDURE. THIS CAN BE A CLUE TO THIS TYPE OF PROTECTION.

ADDITIONAL INFORMATION:
ON SOME DOS 3.3 DISKETTES, THE GAPS BETWEEN THE SECTORS ARE REDUCED IN SIZE. IN SOME CASES THEY CAN BE AS SMALL AS 4 OR 5 BYTES. WHEN NIBBLES AWAY][FINDS THE BEGINNING OF A SECTION OF DATA, IT NORMALLY ADDS 8 BYTES OF SYNC JUST BEFORE THE DATA. THIS WILL NORMALLY PUT SYNC BYTES INTO THE GAP BEFORE THE DATA, WHERE IT SHOULD BE. HOWEVER, IF A DISK HAS VERY SMALL GAPS, THEN THE ADDED SYNC CAN OVERWRITE THE END OF THE PREVIOUS SECTOR. THE PARAMETER FIX AMNT CONTROLS THE NUMBER OF SYNC BYTES WHICH ARE ADDED, SO THIS VALUE CAN BE REDUCED TO PREVENT ANY DATA FROM BEING OVERWRITTEN. THE VALUE THAT NIBBLES AWAY][USES FOR THE SYNC WHICH IT PUTS IN IS CONTAINED IN THE PARAMETER FIX VALU. NORMALLY THIS IS A $7F, BUT IT CAN BE SET TO ANY DESIRED VALUE.
IT SHOULD BE NOTED THAT NIBBLES AWAY][REGARDS ANY DATA BYTE WHICH HAS ITS HIGH BIT CLEARED TO BE A SYNC BYTE. SO THE $7F WHICH IS NORMALLY IN THIS PARAMETER MEANS THAT A SYNC $FF IS TO BE ADDED. IF THE 'OVERIDE STANDARDIZER' OPTION IS SELECTED, THEN NIBBLES AWAY][WILL NOT ADD ANY BYTES, IT WILL SIMPLY CONVERT THE DATA WHICH IS PRESENT BEFORE A SECTOR INTO SYNC, WITHOUT CHANGING ITS VALUE. THIS TECHNIQUE CAN ALSO BE USED FOR DISKS WHOSE GAPS ARE VERY SMALL.

ANOTHER ITEM TO WATCH FOR IS DISKS WHOSE TRACKS APPEAR TO BE VERY LONG. SOME DISK PROTECTION SCHEMES PUT GARBAGE ON A PORTION OF THE TRACK. WHEN THIS GARBAGE IS READ BACK, MORE BYTES ARE READ IN THAN WERE WRITTEN OUT. THIS CAUSES THE TRACK TO BE LONGER THAN NORMAL, AND IN SOME CASES IT BECOMES SO LONG THAT THE DEFAULT PARAMETERS FOR NIBBLES AWAY][CANNOT FIND THE DATA PROPERLY. THE PARAMETERS DATA MIN AND DATA MAX CONTROL THE MINUMUM AND MAXIMUM TRACK LENGTHS (IN INCREMENTS OF 256 BYTES) WHICH NIBBLES AWAY][WILL ACCOMODATE. THE NORMAL VALUE OF DATA MAX IS $1D, BUT THIS CAN BE SET TO A HIGHER VALUE, SUCH AS $25, IF A TRACK APPEARS TO BE VERY LONG. EVEN THOUGH THE TRACK MAY READ IN AS A LARGE NUMBER OF BYTES, MANY OF THESE WILL BE REMOVED BY THE NIBBLE FILTER, SINCE THEY ARE GARBAGE BYTES. THIS WILL ASSURE THAT THE AMOUNT OF DATA WRITTEN BACK OUT WILL NOT BE TO LARGE TO FIT ON THE DESTINATION TRACK.

WHEN NIBBLES AWAY][FINDS A SECTOR OF DATA, IT LOOKS FORWARD IN THE DATA TO FIND A SECOND OCCURANCE OF THE SAME PATTERN. THIS INSURES THAT THE SECTOR HAS BEEN READ IN AND LOCATED CORRECTLY. ON MANY DISKS, THERE IS A PRIMARY SECTION OF DATA, CALLED THE ADDRESS FIELD, AND THE THE ACTUAL DATA FIELD FOLLOWS. IN BETWEEN THESE IS A SMALL GAP, AND MANY TIMES IT CONTAINS RANDOM INFORMATION. THIS MEANS THAT NIBBLES AWAY][SHOULD ONLY MATCH THE NUMBER OF BYTES WHICH ARE FOUND IN THE ADDRESS FIELD, SINCE THE BYTES IN THE GAP MAY NOT READ AS THE SAME VALUE EVERY TIME. THE PARAMETER FIND MAX CONTROLS THE NUMBER OF BYTES WHICH ARE CHECKED DURING THIS PROCEDURE. THE DEFAULT VALUE OF $0C WORKS IN MOST CASES, BUT SOME DISKS USE A SMALLER ADDRESS FIELD WHICH MAY REQUIRE THIS PARAMETER TO BE SET TO A SMALLER VALUE. HOWEVER, IF THIS PARAMETER IS SET TOO LOW, THEN NIBBLES AWAY][MAY IDENTIFY THE MATCH FOR A SECTION OF DATA WHOSE FIRST FEW BYTES ARE THE SAME, BUT WHICH DIFFER LATER ON. THEREFORE ONE SHOULD EXCERSIZE CAUTION WHEN LOWERING THIS VALUE.

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK,HERE TIS SOME BASIC SATELITE TELCO
TUTORIALS NEVER BEFORE WRITTEN!

FIRST OF ALL EVERY SATELITE HAS 24
TRANSPONDERS EACH 36 MHZ WIDE.
INDIVIDUAL TELCO CARRIERS ARE 4KHZ
WIDE.THE VOICE/DATA CARRIER IS USED
TO MODULATE A DOUBLE BALANCED MODULATOR
WHERE ONE OF THE 2 SIDEBANDS TIS ELIMIN
ATED WITH A FILTER.THE REMAINING SIDE
BAND SIGNAL IS APPLIED TO ANOTHER
CARRIER FREQUENCY BETWEEN 64-108 KHZ.
 THESE CARRIERS ARE THEN MULTIPLEXED
TOGETHER IN GROUPS OF 12.SUPERGROUPS
CONTAIN 5 GROUPS AND MASTERGROUPS
CONTAIN 5 SUPERGROUPS.(300 CARRIERS)
 THESE ARE THEN SENT VIA SATELITE IN
"PACKETS" CONTAINING EITHER GROUPS,
SUPERGROUPS,OR MASTERGROUPS IN THE
0 TO 10.75 MHZ RANGE ON A TRANSPONDER.
MASTERGROUPS ARE 5 SUPERGROUPS MULTIPLE
XED AND 1 MIXING CARRIER PER SUPERGROUP
 WHICH ARE UPLINKED BY THE TOC(TOLL
OPERATIONS CENTER) LOCATED IN VARIOUS
AREAS OF THE U.S.
 BLOCK CONVERSION IS USED TO EXTRACT
GROUPS DURING DOWNLINKING.

CONTINUED NEXT MSG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THEN THE GROUP IS MICROWAVED VIA
TERRESTIAL MICROWAVE CIRCUIT TO THE
DESTINATION TOC WHICH DEMODULATES THE
GROUP USING A LOWER SIDEBAND RECEIVER.
THE CARRIERS ARE THEN SENT TO THEIR
FINAL DESTINATION VIA LEASED TELCO
LINE OR RADIO CIRCUIT.

SCPC (SINGLE CHANNEL PER CARRIER)
MAY OPERATE BY THEMSELVES OR BE SLTTED
(OOPS)SLOTTED NEXT TO GROUPS.THESE ARE
60 KHZ WIDE WITHIN 65 TO 85 MHZ

AS SMALL AS AN 4.5 METER DISH WITH 30-
100 WATTS POWER WILL ACHIEVE UPLINK
CAPABILITIES.

TVRO RECIEVES 3.7-4.2 GHZ AND DOWNCONVE
RTS TO SOME IF(SUCH AS 70MHZ) THEN
DEMODULATES TO 0-10.75 MHZ(BASEBAND)
IF YA OWN A TVRO-RUN CABLE FROM THE
VIDEO/DEMODULATED/BASEBAND OUTPUT OF
THE RECEIVER TO A HAM RECEIVER (SUCH
AS AN ICOM R-71A) TO TUNE IN THE
0-10.75 MHZ RANGE OF YOUR SATELITE

OF CHOICE.HEE-HEE-HEE

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK-NOW YA GOTTA FIND OUT WHERE TO LOOK
FOR TELCO TRANSPONDERS.BELOW TIS YE
MAIN SATELITE/TELCO INFO-

SATELITE	TELCO TRANSPONDERS
SATCOM V	3-5-7-11-17-13
SATCOM IR	5-7-9-10-17-23
COMSTAR IV	1-3-4-6-7-15-16-19-22-23
WESTAR IV	14-20-24
TELSTAR IIIA	ALL
COMSTAR III	2-5-6-7-9-14-15-16 18-20-21-22-23
WESTAR II	1-4-5-8-9
GALAXY II	12 MCI TRANSPONDERS HERE
COMSTAR 01/02	ALL
SATCOM IIR	3-4-7-19-21-22-23-

REMEMBER EACH CARRIER MAY USE TO 2700
VOICE CHANNELS WITH NUMBERS GROWING.
DUPLEX FM OR SSB/SCPC CARRIERS ARE
YE FUTURE PHREAKERS TARGETS.

TRYING TO TRACE YE NEW GENERATION OF
SATELITE PHREAKS WILL LEAD TELCO SECUR
ITY STRAIGHT TO A LOCATION IN OUTER
SPACE!!! HAR-HAR-HAR

NUFF SAID-
BOOTLEG

DUE TO POPULAR DEMAND AND THE HUGE
AMOUNT OF INFORMATION NOW AVAILABLE
TO ME,I HAVE DECIDED TO PUBLISH A
SISTER MAGAZINE TO THE BOOTLEGGER
CALLED-

"THE HACKER"

SAME SUBSCRIPTION PRICE AS THE
BOOTLEGGER.SAME ADDRESS ALSO,BUT
THE HACKER WILL BE PUBLISHED IN
BETWEEN BOOTLEGGER ISSUES SO THAT
YOU CAN GET INFO A LOT QUICKER!
NATURALLY THE HACKER WILL PUBLISH
A LOT OF GREAT INFO PERTAINING TO
THE UNDERGROUND HACKING WORLD-
SUBSCRIBE NOW-DON'T MISS ISSUE #1.

(SOME OF THE HACKERS INFO WILL

```
INCLUDE FILES TAKEN RIGHT OUT OF
THE LATEST ESS MANUALS!)

NUFF SAID-
BOOTLEG
-------------------------------------
         FUN STUFF FOR SYSOPS
---------------------------------------
First, you must be a sysop.
(Obviously!) Or, you may be at a
sysop's house (When he or she is not
around.)

Second, you must be VERY popular,
or VERY daring. Either way, your
victim will have a strong dendency
to: a) crash your board, b) hate you,
or c) spread malicious rumors about
you, and, or your board to everyone
in the world that will listen.

    I am going to write about AE
fun first, and then Net-Worx.


                AE Fun
                -- ---

    So you are bored, and want to
have some fun, huh?
    Go into your room, or wherever
you have your apple, and sit down.
Turn on the monitor, and lets see
if there is a leech on the line.
(-note: if you are the unlucky type,
I suggest that you give this up,
because for all you know, that sysop
of the 20meg board is on your line,
and he's going to be your victim!!)
   Now for some of these pranks, you
will need to make things before-
hand. I suggest you read this through,
and make the necessary mods.

1) This one is probably my meanest
trick, and should only be used on
people like Matt Ackerett, or Little
Al.
   Your victim has to be leeching
a game off of your AE for this to
work.
   You wait until your victim is at
his last 2 blocks of memeory to go
until the transfer is done, and
you take out the disk.
   This will ruin the >entire<
transmission. It won't piss them off
too bad if it is only 50 or so blocks,
```

but can you imagine:

```
Send: Matt Ackerett is a fag
290 blocks
crc=167
<289>
```

    Note- The victim has to get 290
blocks, you only let the victim get
289!

   At that point, take out the disk!
They have just waited 1/2 hour for
nothing! They can't get the last block
and have to go through the whole
thing again!! Ha ha!
   This is very mean, especially if
they aren't phreaking, they have
been >paying< for it all!

2) If you want to see if the person
on is intelligent...simply let him
catalog your drive once, then when he
is done, take the disk out, and
put in the disk from the other drive.
When they catalog the disk next, it
will be different!
   This will freak them out, they will
think that they have switched to d1
somehow. The victim will then L)og the
drive, and find it still on D2. Wow!
   Hopefully they will catalog D1
anyway, thinking that they were
originally on D1 and it switched.
Now comes the fun.
   Put the right disk back in D2,
and put the disk that used to be in D2
into D1, so they will get the same
catalog.
   Now they are confused. Now
they will catalog D2, and find the
normal stuff. Hopefull they will
read something, now take the disk
out while they are typing in the name,
and slip the other disk in. It
will say 'file not found.'
   Good. Now they will catalog it,
and look! The wares have changed!
Now something is wrong here! They
will say:

hey! stop it!

   Oh no! They are on to your scheme!
But, 1 last joke! Get a copy-protected
type disk, one that you <gasp> bought.

```
They won't be able to catalog this
at all! Ha!
   If they get mad, they might
say something like:

        Hey! Stop it!

   But will you listen? nnnoooooo!
Take the disk out, and slip something
totally new, preferably the disk that
has "sneakers" or some ancient wares.
Maybe they will think these are the
latest! Watch them post!:

      Hey! I just got some new
      Warez! Do you want to trade??

hah hah!

  Satisfied, you may put the normal
disks back in and walk off to see
some football game.

3) Lock out the space-bar. This will
make it so that they can't type a
<space>. Then, they can't read
anything that requires a space.
Most likely the victim will think that
there is something wrong with >his<
computer. Thusly sending him/her/it
into a 1/2 hour scan of their install
program to see what is wrong.

4) Change the commands...such as:

d)irectory= c)irectory
-           -

   They will have to hack at the
commands! This won't be too funny,
because they won't do anything stupid
like posting:

     hey your commands are screwed!

   Most likely they wont find the
command for 'copy'.


5) lock out the "ctrl-c". This will
piss them off when the victim just
can't exit from posting. Ha!

6) Change the ring count, most, or
almost >all< AE lines are set to
pick up after just 1 ring. Change it
to...say...5 rings, and only tell your
friends that it is at 5 rings. When
```

they call, they will only wait for
about 2 rings, and hang up thinking
that the line is down. Only the people
you like will get on, because they
will be the only ones to wait 5 rings.
Mean huh?

7) When someone is posting, or c)opy-
ing a message, pick up the voice line,
and blow into the reciever. This will
put all of these weird characters onto
the screen. He will save a gay looking
message, that will make it look like
the victim can't type!!

Net Works

--- -----

I don't have as many fun tricks
with net-worx as I do with AE, but
here are a couple of my favorites...

1) In the program, make a bug, like
"ctrl-k" that when pushed (like ctrl-t
for chat) it will dump you into basic.
take out the disks, and put in like
the "bare-bones" net-worx disk and let
them have fun reading fake messages,
mail, and passwords. Ooooh! They will
think:

oh yay! I have everyone's pass!

Now, see if he/she will init the
disks, if they do, you know what type
of user it is. If they are nice,
and 'hang' the line for you so that
no one wil be able to get on after,
or they try to beep you, then give
them a level raise.

2) Be a tyrant. Juggle their levels
while they are on. Like break into
chat, change their level, and watch
them get all mad.

3) Break into chat, and just walk off,
leaving a frustrated user sitting
there.

4) break into chat, and change the
time. In other words, leave them
with -10 minutes, instead of 35 or
so.

5) when they log off, and they get
that stupid message about:

```
            Thank you for
              calling

and all of that, press 'ctrl-c' a
few times, and they will be brought
back. Wow! What happened? Let them
try to log off a few times nd keep
pressing ctrl-c. Finally they should
just press 'reset'. He he!


    I hope you have enjoyed these
little pranks. Your users will hate
you if you do this too often, unless
they are like Matt Ackeret or Little
Al. Then it doesn't matter much.
    Remember! I hold no responsibilty
for people wanting to crash your
system because they are so pissed
at you!

Sysop fun- A Surf Rat file.

Call The Realm of the Rogues!
             415/941-1990  20 megs!!

Call The Twilight Zone!
             408/253-2140  C00L!

Call The Gossip Line! (AE)
             415/949-1049:pw/gossip


And hey! dont put >your< name in here!

Surf..
 -BFB

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

A LOT OF YOU HAVE BEEN ASKING FOR
PROGRAMS THAT WILL HACK OUT VARIOUS
CODES,NUMBERS,PSWDS,ETC.

OK-IVE COMPILED THE MOST POPULAR
AND EFFICIENT HACKING PROGRAMS
EVER ASSEMBLED!THESE INCLUDE SUCH
INFAMOUS PROGRAMS AS THE OUTLAWED
"TSPS" AND THE NOTORIOUS "JOSHUA".
ALONG WITH THESE FAVORATES,INCLUDED
ARE THE 600 CODE PER NIGHT HACKING
PROGRAM BY THE PROFESSOR.ALSO,ALL THE
OTHER UNDERGROUND HACKING PROGRAMS
THAT HAVE EARNED THEIR FAME IN THE
SPIRIT OF WARGAMES!!!

TO ORDER "THE HACKER" SEND $100
```

```
TO-

          THE HACKER

        1080 HAYS CUT-OFF ROAD
        CAVE JCT.OR.97523

NUFF SAID-
BOOTLEG

P.S. THIS COLLECTION OF HACKING
     PROGRAMS WILL DEFINATELY TAKE
     UP SEVERAL DISKS OF SPACE!
THE BOOTLEGGER HAS A FOOLPROOF METHOD
OF SAFELY TRADING DISKS WITHOUT BEING
RIPPED OFF!
SIMPLY SEND 10 OR MORE DISKS TO ME
WITH $2 TO COVER POSTAGE,AND I WILL
HOLD THEM UNTILL THE PERSON YOU ARE TRADING WITH ALSO SENDS THE DISKS YOU
WANTED! WHEN BOTH PARCELS ARE RECEIVED-I'LL
MAIL THEM OUT.IF ONLY ONE PARCEL IS RECEIVED- AFTER 2 WEEKS ILL MAIL IT BACK,OR
 FILL YOUR DISKS WITH NEW PROGRAMS!
I RESERVE THE RIGHT TO COPY ANY PROGRAMS WHILE WAITING! HEE-HEE

NUFF SAID-
BOOTLEG

P.S. AT LEAST ONE PARTY TO THE TRADE
     MUST BE A CURRENT SUBSCRIBER!
ALSO-FILL BOTH SIDES OF YOUR DISKS.
I'VE BEEN GETTING SOME OLD STUFF
IN THE TRADE CLUB LATELY,SO WHAT IM
DOING IS EXCHANGING OLD FOR OLD,NEW
FOR NEW! (GET THE HINT?)

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

VA (VARIABLE ANI ROUTE TREATMENT) IS
USED TO PROVIDE THE START SIGNALS AND
CATAGORY SIGNALS AS REQUIRED FOR
VARIOUS PULSING FORMATS,SUCH AS BELL
SYSTEM STANDARD AND NT-500.THE SYSTEM
OUTPUT AND INPUT PARMS FOR THIS ROUTE
TREATMENT ARE-

ANIFST & ONIST

 START SIGNALS FOR AN ANI/ONI FAIL TYPE
CALL ARE 15 FOR KP,12 FOR ST,13 FOR
STP,14 FOR ST2P,11 FOR ST3P,OR 0 FOR
SENDING THE START SIGNAL PASSED BY THE
TRANSLATOR.

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
```

DATE POSTED:

WANT DTMF DECODER FER YER COMPUTER?

THEY CAN BE HAD FROM $22.95 TO $89.95
FROM ENGINEERING CONSULTING AT
714-671-2009

LOTS OF PHUN WITH YE STUFF THIS COMPANY
SELLS.ASK FOR CATALOG

OH YEA- VISA AND MASTERCARD ACCEPTED!

HAR-HAR-HAR

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK,HERE TIS SOME BASIC SATELITE TELCO
TUTORIALS NEVER BEFORE WRITTEN!

FIRST OF ALL EVERY SATELITE HAS 24
TRANSPONDERS EACH 36 MHZ WIDE.
INDIVIDUAL TELCO CARRIERS ARE 4KHZ
WIDE.THE VOICE/DATA CARRIER IS USED
TO MODULATE A DOUBLE BALANCED MODULATOR
WHERE ONE OF THE 2 SIDEBANDS TIS ELIMIN
ATED WITH A FILTER.THE REMAINING SIDE
BAND SIGNAL IS APPLIED TO ANOTHER
CARRIER FREQUENCY BETWEEN 64-108 KHZ.
THESE CARRIERS ARE THEN MULTIPLEXED
TOGETHER IN GROUPS OF 12.SUPERGROUPS
CONTAIN 5 GROUPS AND MASTERGROUPS
CONTAIN 5 SUPERGROUPS.(300 CARRIERS)
THESE ARE THEN SENT VIA SATELITE IN
"PACKETS" CONTAINING EITHER GROUPS,
SUPERGROUPS,OR MASTERGROUPS IN THE
0 TO 10.75 MHZ RANGE ON A TRANSPONDER.
MASTERGROUPS ARE 5 SUPERGROUPS MULTIPLE
XED AND 1 MIXING CARRIER PER SUPERGROUP
WHICH ARE UPLINKED BY THE TOC(TOLL
OPERATIONS CENTER) LOCATED IN VARIOUS
AREAS OF THE U.S.
BLOCK CONVERSION IS USED TO EXTRACT
GROUPS DURING DOWNLINKING.

CONTINUED NEXT MSG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THEN THE GROUP IS MICROWAVED VIA
TERRESTIAL MICROWAVE CIRCUIT TO THE

DESTINATION TOC WHICH DEMODULATES THE
GROUP USING A LOWER SIDEBAND RECEIVER.
THE CARRIERS ARE THEN SENT TO THEIR
FINAL DESTINATION VIA LEASED TELCO
LINE OR RADIO CIRCUIT.

SCPC (SINGLE CHANNEL PER CARRIER)
MAY OPERATE BY THEMSELVES OR BE SLTTED
(OOPS)SLOTTED NEXT TO GROUPS.THESE ARE
60 KHZ WIDE WITHIN 65 TO 85 MHZ

AS SMALL AS AN 4.5 METER DISH WITH 30-
100 WATTS POWER WILL ACHIEVE UPLINK
CAPABILITIES.

TVRO RECIEVES 3.7-4.2 GHZ AND DOWNCONVE
RTS TO SOME IF(SUCH AS 70MHZ) THEN
DEMODULATES TO 0-10.75 MHZ(BASEBAND)
IF YA OWN A TVRO-RUN CABLE FROM THE
VIDEO/DEMODULATED/BASEBAND OUTPUT OF
THE RECEIVER TO A HAM RECEIVER (SUCH
AS AN ICOM R-71A) TO TUNE IN THE
0-10.75 MHZ RANGE OF YOUR SATELITE
OF CHOICE.HEE-HEE-HEE

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK-NOW YA GOTTA FIND OUT WHERE TO LOOK
FOR TELCO TRANSPONDERS.BELOW TIS YE
MAIN SATELITE/TELCO INFO-

SATELITE	TELCO TRANSPONDERS
SATCOM V	3-5-7-11-17-13
SATCOM IR	5-7-9-10-17-23
COMSTAR IV	1-3-4-6-7-15-16-19-22-23
WESTAR IV	14-20-24
TELSTAR IIIA	ALL
COMSTAR III	2-5-6-7-9-14-15-16 18-20-21-22-23
WESTAR II	1-4-5-8-9
GALAXY II	12 MCI TRANSPONDERS HERE
COMSTAR 01/02	ALL
SATCOM IIR	3-4-7-19-21-22-23-

REMEMBER EACH CARRIER MAY USE TO 2700
VOICE CHANNELS WITH NUMBERS GROWING.
DUPLEX FM OR SSB/SCPC CARRIERS ARE
YE FUTURE PHREAKERS TARGETS.

TRYING TO TRACE YE NEW GENERATION OF
SATELITE PHREAKS WILL LEAD TELCO SECUR
ITY STRAIGHT TO A LOCATION IN OUTER
SPACE!!! HAR-HAR-HAR

NUFF SAID-
BOOTLEG

DUE TO POPULAR DEMAND AND THE HUGE
AMOUNT OF INFORMATION NOW AVAILABLE
TO ME,I HAVE DECIDED TO PUBLISH A
SISTER MAGAZINE TO THE BOOTLEGGER
CALLED-

"THE HACKER"

SAME SUBSCRIPTION PRICE AS THE
BOOTLEGGER.SAME ADDRESS ALSO,BUT
THE HACKER WILL BE PUBLISHED IN
BETWEEN BOOTLEGGER ISSUES SO THAT
YOU CAN GET INFO A LOT QUICKER!
NATURALLY THE HACKER WILL PUBLISH
A LOT OF GREAT INFO PERTAINING TO
THE UNDERGROUND HACKING WORLD-
SUBSCRIBE NOW-DON'T MISS ISSUE #1.

(SOME OF THE HACKERS INFO WILL
INCLUDE FILES TAKEN RIGHT OUT OF
THE LATEST ESS MANUALS!)

NUFF SAID-
BOOTLEG

```
=============================================================================
DOCUMENT bootl-6.hac
=============================================================================

---------------------------------------
--  How to modify the 16k Ram Board  --
             By: Axe Man
---------------------------------------

WRITE PROTECT:
    LIFT PIN #3 FROM U18 CHIP & CONNECT
    TO ONE SIDE OF SWITCH.
    CONNECT SOCKET AND PIN #13 74LS175
    TO CENTER OF SWITCH
    CONNECT TOP OF R3 TO OTHER SIDE OF
    THE SWITCH

R3---------------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #13----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------0
U18

CHANGES FOR RAM & ROM
    LIFT PIN #3 FROM U14 CHIP & CONNECT
    TO ONE SIDE OF SWITCH
    CONNECT SOCKET AND PIN #5 74LS175
    TO CENTER OF SWITCH
    CONNECT GROUND TO OTHER SIDE

GROUND-----------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #5-----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------O
U14

* * * * * * * W A R N I N G * * * * * *
THIS IS DONE AT YOUR OWN RISK
IT WILL VOID YOUR GUARANTEE
WE ASSUME NO RESPONSIBILITY FOR RESULTS
* * * * * * * W A R N I N G * * * * * *

IT SEEMS THERE'S A DEMAND FOR A W/P
SWITCH ON THE ANDROMEDA -- SO HERE IT
IS ...

 LOCATED ON THE ANDROMEDA RAM CARD IS
 A PIN NUMBER 25 WHICH HAPPENS TO BE
 THE POWER (+5V) PIN. IF THIS PIN IS
 FOLLOWED ONTO THE PC BOARD, THERE WILL
 BE TWO RESISTORS (SMALL TUBE-LIKE
THINGS WITH COLOR BANDS AND ONE LEAD
OUT OF EACH END). AT ONE END THE POWER
WILL GO INTO THIS RESISTOR, AT THE OTHE
R ANOTHER TRACE WILL GO OFF TO SOME
OF THE OTHER ELECTRONICS ON THE BOARD.
WE WANT TO USE THE END THAT HAS THE
TRACES GOING TO OTHER CHIPS ON THE
BOARD. (CALL THIS POINT #1 (USE EITHER
RESISTOR - THERE ARE TWO)). POINT NUMBE
R TWO IS WHERE PIN 18 FROM THE APPLE
CONNECTOR (7 PINS DOWN FROM 25 ON THE
SAME SIDE) ENTERS ONTO THE PC BOARD
AND IMMEDIATELY GOES THROUGH TO THE
OTHER SIDE (AFTER ABT 1/2 "). THIS
IS POINT #2.  IF YOU TRACE WHERE TH
E THING COMES OUT ON THE OTHER SIDE,
YOU'LL FIND OUT THAT IT POPS BACK ON
THE SIDE IT STARTED FROM ABOUT 1/2"
LATER... THIS LITTLE LINK IS WHEERE WE
CUT THE TRACE TO INSERT THE SWITCH.
OK, WE CUT THE TRACE BETWEEN THE TWO
POINTS THAT IT GOES THROUGH THE PC
BOARD. LABEL THE OTHER PLACE WHERE THE
TRACE GOES THROUGH POINT#3.  NOW WE
WILL ATTACH AN SPDT SWITCH TO THE BOARD
 SOLDER ONE WIRE TO POINT 3, AND ATTACH
IT TO THE CENTER TERMINAL OF THE SWITCH
THEN SOLDER A WIRE TO POINT 1 AND
ATTACH IT TO EITHER SIDE OF THE CENTER
SWITCH. LASTLY, TAKE A WIRE AND SOLDER
IT TO POINT 2 AND THEN TO THE UNUSED
PIN ON THE SWITCH. THERE YOU HAVE IT!
WHEN THE SWITCH HANDLE IS ON THE SAME
SIDE AS THE WIRE FROM POINT #1, REG-
ULAR OPERATION WILL TAKE PLACE. IF THE
SWITCH IS THROWN IN THE OTHER DIRECTION
THE CARD WILL BE WRITE PROTECTED.
(*PLEASE NOTE THAT THIS MODIFICATION
 WILL VOID YOUR WARRANTY AND THAT THE
USER ASSUMES AND WILL BE RESPONSIBLE
FOR ALL RISKS AND DAMAGES INCURRED IN
THE MAKING OR THE USE OF THIS MOD-
IFICATION, AND THAT THIS MODIFICATION
IS NOT GUARANTEED TO BE SUITABLE FOR
ANY PARTICULAR PURPOSE*)
----------------------------------------

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

FOR YE WITH SCANNERS IN NEW JERSEY-
N.J. BELL SECURITY CAN BE FOUND AT

462.55   462.575   462.600   462.625
```

```
462.65   462.675   462.700   462.725

ABOVE FREQUENCYS ARE IN MHZ.

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THATS RIGHT! IF YA WANT A CATALOG OF YE
BELL DOCUMENTATION CATALOG FER FREE-
THEN CALL 1-800-432-6600

THIS CATALOG LISTS AT&T DOCS FOR SALE!

IF YA GET ANY-LETS TRADE-CALL ME VIA
VOICE AT 503-592-4461

NUFF SAID-
BOOTLEG

OK-TO ALL OF YOU FOOLISH ENOUGH TO
ASK

     NO-NO-NO FREE SAMPLE ISSUES!

BUT BACK ISSUES ARE AVAILABLE FOR
$10 EACH.

IF YOU HAVE PROBLEMS WITH ANY DISK
NOT WORKING-MAIL IT BACK WITH AN
EXPLANATION OF THE PROBLEM.
ILL REMAIL A FRESH COPY UPON GETTING
THE OLD ONE.

AND TELL YOUR FRIENDS-NO FREE SAMPLES
(UNLESS THEY FEEL LIKE MAILING FREE
SAMPLES OF 10 DOLLAR BILLS FIRST!)

NUFF SAID-
BOOTLEG

MSG LEFT BY: SALLY RIDE
DATE POSTED:

AN INTERESTING MESSAGE POSTED ON TEXCON
CAUGHT MY EYE..IN LIGHT OF THE NAZI BBS
THE MOLESTERS BBS AND THE BUST IN N.J..
A NEW FEDERAL LAW THAT WOULD OUTLAW OR
SEVERELY RESTRICT BBS ACTIVITY IS BEING
PUSHED THROUGH CONGRESS, ACCORDING TO
TWO SOURCES THE METAL DETECTOR AND JOHN
EDENS. ANOMG THE RESTRICTIONS DISCUSSED
ARE: *REGISTRATION OF ALL BBS AS PUBLIC
      UTILITIES.
     *SYSOPS REQUIRED TO KEEP LOGS OF
      ALL USERS "VERIFIED" NAMES AND
      ADDRESSES.
     *SYSOPS REQUIRED TO KEEP LOGS OF
      ALL MESSAGE POSTINGS AND TIMES OF
      POSITNG
     *CRIMINAL PENALTIES FOR SYSOPS WHO
      ALLOW ILLEGAL MESSAGE POSTINGS
      WHETHER OR NOT THEY HAVE KNOW OF
      THE CONTENT OR HAVE HAD A CHANCE
      TO REMOVE IT.
     *BBS USERS WOULD BE "REQUIRED" TO
      USE THEIR LEGAL NAMES.
I HAVE NO WAY TO BE SURE THIS DATA IS
CORRECT, BUT I SUSPECT IT IS CLOSE TO
THE TRUTH. CONTACT YOUR CONGRESSIONAL
REPRESENTATIVE, THEY ALMOST ALL HAVE
800 #'S, AND FIND OUT AND EXPRESS YOUR
OPINION, TOO, WHILE YOU HAVE THE CHANCE
SALLY RIDE:::SPACE CADET

MSG LEFT BY: THE WARLOCK LORD
DATE POSTED:

IS SO DAMN UNCONSTITUTIONAL IT MAKES ME SICK.  THE SYSOP SHOULD NOT BE HELD
RESPONSIBLE FOR MESSAGES, HE IS PROTECTED BY THE FIRST AMENDMENT.  IT'LL
NEVER PASS.

THE WARLOCK LORD
WHEW-

WHAT A YEAR SO FAR-
BOARDS BEING BUSTED THROUGHOUT THE U.S.
NEW COMPUTER CRIME LAWS BEING ABUSED BY
COPS IN MOST STATES (THOUGH MOST OF THESE LAWS CERTAINLY WILL PROVE
UN-CONSTITUTIONAL IF EVER TAKEN TO THE SUPREME COURT).
AND-NEW FEDERAL LAWS IN THE FEDERAL SUB-COMMITTEES WAITIN TO BE VOTED ON!
ALSO,YE SECRET SERVICE HAS TAKEN OVER
FEDERAL COMPUTER INVESTIGATIONS FROM
THE FBI.

SEEMS LIKE EVERY PREDICTION I MADE IN
EARLIER ISSUES HAS COME TRUE!

NOW LET ME TELL YA WHATS HAPPENIN AND
WHY. THE FEDS KNOW EVERYTHING IS GOIN
COMPUTER IN THE NEAR FUTURE,AND NATURALLY LAW ENFORCEMENT AGENCYS WANTED THE
WORST LAWS POSSIBLE ON THE BOOKS SO
THEY COULD RUN AMOK WITHOUT ANY CONCERN
OF OUR CONSTITUTIONAL RIGHTS.
SINCE OUR LEGISLATORS KNOW LITTLE OF
COMPUTER HAPPENINGS,YE SNEAKY FEDS
DECIDED TO RAID VIRTUALLY EVERY KID
THAT RAN A GOOD BOARD THIS SPRING.
NATURALLY,THEY SPENT MEGA-THOUSANDS
ON THESE RAIDS AND THEN WERE KIND
ENOUGH TO MIS-INFORM THE PRESS AS
TO FABLES LIKE HACKERS MOVIN SATELITES,
ETC.NOW WHEN OUR INK HUNGRY MEDIA GOT
HOLD OF ALL THESE LIES, THEY TRUMP
```

```
EM UP EVEN FURTHER IN A BLITZ THAT
INCLUDED FRONT PAGE HEADLINES,UPI,
AND NATIONAL TV.
 O.K. SO HERE SITS MR & MRS CITIZEN
BELIEVIN THAT GARBAGE AN CALLIN MR
CONGRESSMAN SCREAMIN BLOODY MURDER.
 AN YA CAN GUESS WHATS HAPPENIN NOW-
YEP-THEYRE TRYIN TO DEPRIVE US OF
CONSTITUTIONAL RIGHTS (1ST & 4TH
AMMENDMENTS) THROUGH NEW PENDING
LEGISLATION......

NOW YA SAY-WHAT CAN BE DONE?

OK-WRITE EVERY CONGRESSMAN IN YOUR
STATE AND ALSO OTHER STATES AN EVEN
THE PRESIDENT TELLIN THEM YOU DONT
WANT ANY LAWS PASSED THAT WILL
INFRINGE ON YOUR RIGHTS RE COMPUTERS.
MASS PRODUCE THOSE LETTERS & SEND EM
OUT NOW-THESE BAD LAWS CAN ONLY BE
STOPPED BY DOIN SOMETHING RIGHT AWAY.

ALSO-BITCH ABOUT SPENDING ALL OUR
TAX MONEY HARASSING KIDS.THERE IS
A LOT OF CROOKS RUNNIN ROUND THEY
SHOULD BE AFTER WITH THOSE FEDERAL
AGENTS.

AND MAKE UP PETITIONS SIGNED BY
ANYONE FROM ADULTS TO YOUR CLASSMATES
TO SEND IN ALSO. HECK-SEND THOSE
LETTERS IN EVERY WEEK TILL WE GET
RESULTS.ALSO-POST THIS FILE EVERYWHERE.



         THE BOOTLEGGER MAGAZINE
         1080 HAYS CUT-OFF ROAD
         CAVE JCT.OR.97523

NUFF SAID-
BOOTLEG
---------------------------------------
--  How to modify the 16k Ram Board  --
             By: Axe Man
---------------------------------------

WRITE PROTECT:
    LIFT PIN #3 FROM U18 CHIP & CONNECT
    TO ONE SIDE OF SWITCH.
    CONNECT SOCKET AND PIN #13 74LS175
    TO CENTER OF SWITCH
    CONNECT TOP OF R3 TO OTHER SIDE OF
    THE SWITCH

R3--------------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #13----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------0
U18
```

```
CHANGES FOR RAM & ROM
    LIFT PIN #3 FROM U14 CHIP & CONNECT
    TO ONE SIDE OF SWITCH
    CONNECT SOCKET AND PIN #5 74LS175
    TO CENTER OF SWITCH
    CONNECT GROUND TO OTHER SIDE

GROUND-----------------O
                       !
                       /  NORMAL OPEN
                       !
PIN #5-----------------O
74LS175                !
                       /  NORMAL CLOSED
                       !
PIN #3-----------------O
U14

* * * * * * * W A R N I N G * * * * * *
THIS IS DONE AT YOUR OWN RISK
IT WILL VOID YOUR GUARANTEE
WE ASSUME NO RESPONSIBILITY FOR RESULTS
* * * * * * * W A R N I N G * * * * * *

IT SEEMS THERE'S A DEMAND FOR A W/P
SWITCH ON THE ANDROMEDA -- SO HERE IT
IS ...

 LOCATED ON THE ANDROMEDA RAM CARD IS
 A PIN NUMBER 25 WHICH HAPPENS TO BE
 THE POWER (+5V) PIN. IF THIS PIN IS
 FOLLOWED ONTO THE PC BOARD, THERE WILL
 BE TWO RESISTORS (SMALL TUBE-LIKE
THINGS WITH COLOR BANDS AND ONE LEAD
OUT OF EACH END). AT ONE END THE POWER
WILL GO INTO THIS RESISTOR, AT THE OTHE
R ANOTHER TRACE WILL GO OFF TO SOME
OF THE OTHER ELECTRONICS ON THE BOARD.
WE WANT TO USE THE END THAT HAS THE
TRACES GOING TO OTHER CHIPS ON THE
BOARD. (CALL THIS POINT #1 (USE EITHER
RESISTOR - THERE ARE TWO)). POINT NUMBE
R TWO IS WHERE PIN 18 FROM THE APPLE
CONNECTOR (7 PINS DOWN FROM 25 ON THE
SAME SIDE) ENTERS ONTO THE PC BOARD
AND IMMEDIATELY GOES THROUGH TO THE
OTHER SIDE (AFTER ABT 1/2 "). THIS
IS POINT #2.  IF YOU TRACE WHERE TH
E THING COMES OUT ON THE OTHER SIDE,
```

YOU'LL FIND OUT THAT IT POPS BACK ON
THE SIDE IT STARTED FROM ABOUT 1/2"
LATER... THIS LITTLE LINK IS WHEERE WE
CUT THE TRACE TO INSERT THE SWITCH.
OK, WE CUT THE TRACE BETWEEN THE TWO
POINTS THAT IT GOES THROUGH THE PC
BOARD. LABEL THE OTHER PLACE WHERE THE
TRACE GOES THROUGH POINT#3. NOW WE
WILL ATTACH AN SPDT SWITCH TO THE BOARD
 SOLDER ONE WIRE TO POINT 3, AND ATTACH
IT TO THE CENTER TERMINAL OF THE SWITCH
THEN SOLDER A WIRE TO POINT 1 AND
ATTACH IT TO EITHER SIDE OF THE CENTER
SWITCH. LASTLY, TAKE A WIRE AND SOLDER
IT TO POINT 2 AND THEN TO THE UNUSED
PIN ON THE SWITCH. THERE YOU HAVE IT!
WHEN THE SWITCH HANDLE IS ON THE SAME
SIDE AS THE WIRE FROM POINT #1, REG-
ULAR OPERATION WILL TAKE PLACE. IF THE
SWITCH IS THROWN IN THE OTHER DIRECTION
THE CARD WILL BE WRITE PROTECTED.
(*PLEASE NOTE THAT THIS MODIFICATION
 WILL VOID YOUR WARRANTY AND THAT THE
USER ASSUMES AND WILL BE RESPONSIBLE
FOR ALL RISKS AND DAMAGES INCURRED IN
THE MAKING OR THE USE OF THIS MOD-
IFICATION, AND THAT THIS MODIFICATION
IS NOT GUARANTEED TO BE SUITABLE FOR
ANY PARTICULAR PURPOSE*)

COPY II PLUS 4.1 DISK-BACKUP INSTRUCTI
ONS
1/19/83

'T' INDICATES A TRACK NUMBER. WHEN A
RANGE OF TRACKS ARE TO BE COPIED, YOU
WILL SEE "TX-TXX". THIS MEANS SET THE
START TRACK TO "X", AND THE END TRACK
TO "XX". IF ONLY A SINGLE TRACK IS TO
BE COPIED, YOU WILL SEE "TX". THIS
MEANS USE "X" FOR BOTH START AND END
TRACKS. NUMBERS TO THE RIGHT ARE
PARAMETER CHANGES THAT SHOULD BE MADE
BEFORE COPYING THE TRACKS SHOWN.
"STEP" MEANS TRACK INCREMENT.

WHEN MAKING A BACKUP, BE SURE TO FOLLOW
THE STEPS IN ORDER. OFTEN A PARAMETER
WILL NOT BE RE-LISTED IF IT IS SET FOR
A PRIOR RANGE OF TRACKS.

IF A PARAMETER LISTING INCLUDES "SECTOR
EDIT", USE THE COPY II PLUS SECTOR EDIT
OR TO MODIFY THE TRACK AND SECTOR
SHOWN. BE SURE TO PATCH THE READ/WRITE
ROUTINES IF THE LISTING SHOWS "PATCHED"
AND TO USE THE CORRECT DOS (3.2 OR 3.3)

```
.

SOME DISKETTES CAN BE DUPLICATED USING
THE DEFAULT PARAMETERS (OR COPY DISK
FROM THE MAIN MENU).  IF THE DISKETTE
YOU WISH TO BACKUP IS NOT LISTED, TRY
THE DEFAULT SETTINGS OR COPY DISK FIRST
.

A "*" NEXT TO THE PRODUCT NAME IN-
DICATES THESE PARAMETERS WERE USER
SUBMITTED AND HAVE NOT BEEN VERIFIED BY
CENTRAL POINT SOFTWARE.  WE ENCOURAGE
OUR CUSTOMERS TO LET US KNOW WHEN THEY
BACKUP A DISK NOT ON THIS LIST.  THIS
INFORMATION IS MADE AVAILABLE TO ALL
COPY II PLUS OWNERS.

NOTE TO ALL IBM PC OWNERS: COPY II PC
IS NOW AVAILABLE.  SAME PRICE AS COPY
II PLUS FOR THE APPLE, AND IT IS HANDS-
DOWN THE FASTEST, MOST RELIABLE AND
MOST POWERFUL COPY PROGRAM FOR THE
PERSONAL COMPUTER.

ALIEN RAIN & TYPHOON    (BRODERBUND)
        T0-T5           9=0, 31=0, D=D5
, F=0
        T6-TE           E=DE

APPLE ADVENTURE *
        T0-T22          D=1, 10=96, 24=
96

APPLE LOGO *            (APPLE COMPUTER
)
        T0-T22
        T1              A=1, 4B=1, 50=1
  (ERROR 6 OK)

APPLE PANIC *           (BRODERBUND)
        T0-TD

APPLE WORLD *           (USA)
        T0-T23

APPLEWRITER II          (APPLE)
        T0-T22          10=96

APPLEWRITER ///         (APPLE)
        T0-T22          D=1, 10=96, 24=
96

A2-PB1  (PINBALL)       (SUB LOGIC)
        T0              10=96
        T1-T15          A=3, E=DB, F=AB
, 10=BF, 44=1,
                        45=D, 46=F
```

```
AZTEC *
        T0-22           D=1, 10=96, 24=
96

BACK-IT-UP II *         (SENSIBLE)
        T0              10=96, 9=0
        T1.5-TB.5       10=B5, A=3

BEER RUN
        T0              9=0
        T1.5-TD.5       D=1, 3B=40

CANNONBALL BLITZ *
        T0-T22
        T3-TF           3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

CANNONBALL BLITZ (ALTERNATE)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 17, SECTOR E.
          CHANGE ADDRESS CD FROM 49 TO
60

CASTLE WOLFENSTEIN      (MUSE)
        T0-T22          D=1, 31=0

CEILING ZERO *
        T0-T2
        T3-T11          9=0, E=D6, 1C=D
6, 34=1, 38=F9, 4F=1

CHESS 7.0 *             (ODESTA)
        T0-T22          10=96, 9=0

CHOPLIFTER & SERPENTINE (BRODERBUND)
        T0              A=3, 44=1, 45=D
, 9=0, 0=F, 50=3
        T1-T8           4=FD, 31=0, 43=
0, 45=10, 4F=1, 46=12
        T9              45=8, 46=D
        TA-TB           45=2
        TC-T1E.5 STEP .5        45=8, 1
0=D4, 51=1, D=1
        T20             45=6, D=0, 4F=0

NOTE: CHOPLIFTER, SERPENTINE, DAVID'S M
IDNIGHT MAGIC AND STARBLAZER USE TRACK A
RCING AND ARE VERY SENSITIVE TO DRIVE SP
EED.  IF YOU HAVE PROBLEMS, TRY REVERSIN
G DRIVES.

COLOSSAL CAVE ADVENTURE *
        T0-T22

CRANSTON MANOR          (ON-LINE)
```

```
        T0-T22
        T18             3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

CROSSFIRE               (ON-LINE)
        T0-TB           9=0
        T1              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

CRUSH, CRUMBLE AND CHOMP *
        T0-T22          10=96, 9=0

DAVID'S MIDNIGHT MAGIC  (BRODERBUND)
        T0              A=3, 44=1, 45=D
, 9=0, 0=F, 50=3
        T1-TA           44=0
        TB              44=1, 31=0, 43=
0, 45=8
        TC-T19 STEP .5  10=F5, F=FD, 51
=1, 4F=1, D=1
        SEE NOTES FOR CHOPLIFTER

DB MASTER               (STONEWARE)
        T0-T5           10=96, 24=96, D
=1
        T6.5-T22.5      D=0

DEADLINE *              (INFOCOM)
        T0-T22

DESKTOP PLAN II         (VISICORP)
        T0-T22          10=96, 34=1, 36
=2A

DISK ORGANIZER *
        T0
        T1              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)
        T2-T4           D=1
        TA-TB

ELECTRIC DUET *         (INSOFT)
        USE COPY DISK FROM MAIN MENU

ESCAPE *
        T0-T22

EXECUTIVE SECRETARY *
        T0-T22          9=0, 8=1, 10=96

EXPEDITOR               (ON-LINE)
        T0-22           10=96
        T3 & T1F        3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)
```

```
FORMAT II *
        USE COPY DISK FROM MAIN MENU

FS-1 (FLIGHT SIMULATOR) (SUB LOGIC)
        T0                      10=96
        T1.5-T21 STEP 1.5       E=DB, F
=AB, 10=BF, A=3, 4E=1
        T7-T8
        T9.5

GALACTIC GLADIATORS *
        T0-T20          10=B7, E=D7, 9=
0, 31=0
        T21-T22         34=1

GORGON                  (SIRIUS)
        T0              10=96, 9=0
        T1.5-E.5        D=1, 24=96, A=3
, E=DD, F=AD,
                        10=DA, 3B=40

HYPERSPACE WARS *       (CONTINENTAL)
        T0-T22          9=0

JAW BREAKER *           (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

KRELL LOGO *
        T0-T22

LIST HANDLER AND UTILITY *
        T1-T11
        T0              9=0, A=3, 44=1,
 45=D, 50=3
        T12-T22.5 STEP .5       D=1, E=
F5, F=D7, 10=F7
                                45=8, 4
6=D, 51=1
        (SEE NOTES FOR CHOPLIFTER)

MAGIC WINDOW *
        T0-T22

MICRO WAVE *            (CAVALIER)
        T0-T22
        T11             3B=1, A=1, 4B=1
, 4D=8, 50=1

MOUSKATTACK *           (SIERRA ON-LINE
)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 18, SECTOR 3
          CHANGE ADDRESS B1 FROM 49 TO
```

```
60

MULTI PLAN              (MICROSOFT)
        T0-T22          10=96

OLYMPIC DECATHALON *    (MICROSOFT)
        T0-T22          9=0

ORBITRON *
        T0-T1           9=0, 31=0
        T1.5-TF.5
        WRITE PROTECT COPY!

PFS & PFS REPORT        (SOFTWARE PUBLI
SHING CORP.)
        USE "COPY DISK" FROM MAIN MENU.
 AFTER COPYING AND BEFORE
        USING, PUT A TAB OVER THE WRITE
 PROTECT NOTCH OR  THE
        COPY WILL NOT WORK.

PHANTOMS FIVE           (SIRIUS)
        T0              9=0
        T2-T1C          3A=0, 50=20

PRISM *
        T0-T22

PRISONER *
        T0-T22

RASTER BLASTER          (OLD & NEW VERS
IONS - BUDGECO)
        T0              10=96
        T5-T11  STEP 4  D=1, 9=0, 31=0,
 A=2, E=AD,
                        F=DE, 3B=40
        T6-T12  STEP 4
        T7.5-TF.5  STEP 4
        T1.5-T3.5  STEP 2

SABATOGE *
        T0-T22
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

SARGON *                (HAYDEN)
        T0-T1A

SCREENWRITER II *
        COPY DISK, THEN SECTOR EDIT
          DOS 3.3 PATCHED
          TRACK 3, SECTOR B
          CHANGE ADDRESSES 94, 95, 96 T
O EA EA EA

SNOGGLE *               (BRODERBUND)
```

```
        T0-T9           9=0, 8=1

SPACE INVADERS *
        T0-T22          10=96

SNACK ATTACK            (DATA MOST)
        T0-T12
        SECTOR EDIT DOS 3.2 PATCHED
          TRACK 0 SECTOR 3
          CHANGE ADDRESS 63 FROM 38 TO
18

SNEAKERS                (SIRIUS)
        T0              9=0, 10=96, 44=
1, 45=10, D=1
        T1.5-TC.5       44=0
        TD.5            44=1

SOFTPORN ADVENTURE      (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

SPACE EGGS *            (SIRIUS)
        T0              9=0
        T2-T6
        T11-1A

SPACE VIKINGS *
        T0-T22

SPEED READING *
        T0-T22          9=0, 10=96

SPIDER RAID *           (INSOFT)
        T0
        T1-T17          A=3, E=92, F=93
, 4F=1, 10=95, 44=1
                        46=A, 9=0, 8=1,
 D=1, 24=96
                        3F=1, 34=1, 36=
2A, 37=97
                        31=0, 43=0
        T1.5-T17.5      E=95, 10=92
        (SEE NOTES FOR CHOPLIFTER)
        (ONLY WORKS ON NEW VERSIONS)

STARBLASTER *
        T0              10=96, 9=0
        T7-T20 STEP 1.5 E=DF, F=AD, 10=
DE

STARBLAZER              (BRODERBUND)
        SAME AS CHOPLIFTER

STARCROSS *             (INFOCOM)
        T0-T22          10=96
```

```
STELLAR INVADERS *      (APPLE)
        T0-T22

STOCK PORTFOLIO SYSTEM *
        T3-T22
        T0-T2           4=FD, 8=1, 10=A
D

TAX MANAGER *           (MICROLAB)
        USE COPY DISK FROM MAIN MENU

TAX PREPARER *          (HOWARDSOFT)
        USE COPY DISK FROM MAIN MENU

THRESHOLD               (ON-LINE)
        T0-T22
        T1-T23 STEP 22  3B=1, A=1, 4B=1
, 4D=8, 50=1
                        (ERROR 6 OK)

TUBE WAY *
        T0-T22

TYPING TUTOR *          (MICROSOFT)
        USE COPY DISK FROM MAIN MENU

ULTIMA II *
        COPY DISK, THEN SECTOR EDIT
          TRACK 3, SECTOR 0C
          CHANGE ADDRESSES 84, 85, 86 A
LL TO EA.

ULTIMA II *
        T0-T22          10=96, 9=0, 34=
1, 31=0

VERSAFORM *
        T0-T22

VISICALC                (VISICORP)
        T0-T16

VISICALC ///            (APPLE COMPUTER
)
        T0-T22          10=96, 24=96, D
=1

VISIDEX, VISISCHEDULE, VISITERM, VISITR
END/VISIPLOT  (VISICORP)
        DON'T USE BIT COPY.  USE "COPY
DISK" FROM MAIN MENU.

VISIFILE                (VISICORP)
        T0-T22          10=96, 34=1, 36
=2A, 37=EB, 3E=2

WIZARDRY * (FRONT SIDE)
```

```
        COPY DISK THEN USE BIT COPY:
        T3-T23          10=96, 24=96, D
=1
        NOTE: WRITE PROTECT BACKUP OR I
T WILL NOT WORK.

WIZARDRY * (BACK SIDE)
        T1-T22          10=96, 24=96, D
=1

WORD HANDLER *
        USE COPY DISK FROM MAIN MENU

ZARGS *                 (INOSFT)
        SAME AS SPIDER RAID

(?=MENU, 1-9) ->:

---------------------------------------
       COPY II+ PLUS PARAMETERS
---------------------------------------

FROM:
     THE FORBIDDEN ZONE
     THE ROM RAIDER

UPDATE:3/28/83

3-D Graphics System * (Cal Pacific Computer)
     T0-T8
     T11-T12
     T15-T17

3-D Graphics System *
     T0-T2
     T4-T8
     T11-T18

Adventure To Atlantis * (Synergistic)
     T0-T22             10=96 24=96 9=0 31=0 D=1

Air Simulator *         (Mind Systems)
     T0-TF

Air Traffic Controller *
     T0-T22             10=96
     T23                31=0 50=1 10=96

Akalabeth *
     T0                 9=0 31=0
     T2-T3              E=DE F=AA 10=AD
     T6-T18

Alien Rain & Typhoon     Broderbund)
     T0=T5              9=0 31=0 D=D5 F=0
     T6-TE              E=DE

Alkem Stones *
```

```
     T0-T22             A=3 10=96

Apple Adventure *
     T0-T22             D=1 10=96 24=96

Apple Cillin II *
     T0-TC

Apple //e  Business Graphics *
     T0-T22             D=1 10=96 24=96

Apple /// Business Graphics *
     T0-T22  (error 2 OK)

Apple Logo *
     T0-T22
     T1                 A=1 4B=1 50=1 E=FC 19=FD 1C=AA 1F=EE
            or for T1
     T1                 A=1 4B=1 50=1 E=AA 1C=AA
               or
     T1                 A=1 4B=1 50=1 3B=1 4D=8

NOTE:  We have been told that Apple Logo requires persistance!
Keep trying
track 1 until the disk works.

Apple Panic *           (Broderbund)
     T0-TD

Apple Panic *
     T0-T5              9=0 F=0
     T6-TD              E=DE

Apple Pilot and Super Pilot *
     T0-T22

Apple World *           (USA)
     T0-T23

Apple Writer II (and IIe)
     T0-T22             10=96

Apple Writer II Pre-Boot *
     T0-T22             10=96 9=0

Apple Writer ///        (Apple)
     T0-T22             D=1 10=96 24=96

A2-PB1 (Pinball)        (Sub-Logic)
     T0                 10=96
     T1-T15             A=3 E=DB F=AB 10=BF 44=1 45=D 46=F

AZTEC *
     T0-T22             D=1 10=96 24=96

Back-It-Up II *         (Sensible)
     T0                 10=96 9=0
     T1.5-TB.5          10=b5 A=3
```

```
Battle of Shilo *
     T0-T22             E=D4 10=b7

Beer Run
     T0                 9=0
     T1.5-TD.5          D=1 3B=40

Bomb Alley *
     T0-T22             E=D4 10=B7 34=1 37=6E 38=fe

Borg *                  (Sirius)
     T0                 10=96 9=0
     T1.5-TB.5          D=1 24=96 A=3 E=DD F=AD 10=DA 3B=40
     TD-T20

Cannonball Blitz *
     T0-T22
     T3-TF              3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

Cannonball Blitz (alternate)
     T0-T22             10=96
     Sector Edit Dos 3.3 Patched
       Track 17, Sector E
       Change Address CD From 49 to 60

Castle Wolfenstein      (Muse)
     T0-T22             D=1 31=0

Caves of Olympus *
     T0-T22             10=96 9=0

Ceiling Zero *
     T0-T2
     T3-T11             9=0 E=D6 1C=D6 34=1 38=F9 4F=1

Chess 7.0 *             (Odesta)
     T0-T22             10=96 9=0

Chess 7.0 *
     T0-T22             10=96 9=0 8=1 3E=2

Choplifter, Serpentine, & Starblazer (Broderbund)
     T0                 A=3 44=1 45=d 9=0 0=F 50=3
     T1-T8              4=FD 31=0 43=0 45=10 4F=1 46=12
     T9                 45=8 46=D
     TA-TB              45=2
     TC-T1E.5 Step .5   45=8 10=D4 51=1 D=1
     T20                45=6 D=0 4F=0

NOTE:  Choplifter, Serpentine, David's Midnight Magic and
Starblazer use track
arcing and are very sensitive to drive speed. If you have
problems, try
reversing drives.
       -- Sea Fox may also copy using these parameters --

Colossal Cave Adventure *
```

```
     T0-T22

Congo *
     T0-T22            D=1 9=0 24=96 10=96

Cranson Manor          (On-Line)
     T0-T22
     T18               3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

Crossfire              (On-Line)
     T0-T8             9=0
     T1                3B=1 A=1 4B=1 4D=8 50=1 (error 6 OK)

If you want CRUSH, CRUMBLE AND CHOMP you're crazy! That game
sucks!

Dark Crystal           (On-Line)
     Copy all 4 sides from main menu
     Sector Edit side 1A as follows:
       Track 5, Sector F change address A8-AA all to EA
       Track 7, Sec C, change addressess 22-24 all to EA

David's Midnight Magic  (Broderbund)
        T0             A=3, 44=1, 45=D, 9=0, 0=F, 50=3
        T1-TA          44=0
        TB             44=1, 31=0, 43=0, 45=8
        TC-T19 STEP .5  10=F5, F=FD, 51=1, 4F=1, D=1
        SEE NOTES FOR CHOPLIFTER

DB MASTER              (STONEWARE)
        T0-T5          10=96, 24=96, D=1
        T6.5-T22.5     D=0

DEADLINE *             (INFOCOM)
        T0-T22

DESKTOP PLAN II        (VISICORP)
        T0-T22         10=96, 34=1, 36=2A

DISK ORGANIZER *
        T0
        T1             3B=1, A=1, 4B=1, 4D=8, 50=1
                       (ERROR 6 OK)
        T2-T4          D=1
        TA-TB

DLM Software *
        T0-T22

Dragon Fire *
        T0-T22         10=96 9=0

Early Games *
        Use Copy Disk from Main Menu

Education Activities Software *
        T0-T22
```

```
Einstein Computer *
        Copy Disk from Main Menu
        Sector Edit Track 8, Sector 4
        Change Addresses 2A-2C from BD 8C C0 to 4C E2 91

ELECTRIC DUET *         (INSOFT)
        USE COPY DISK FROM MAIN MENU

ESCAPE *
        T0-T22

EXECUTIVE SECRETARY *
        T0-T22          9=0, 8=1, 10=96

EXPEDITOR               (ON-LINE)
        T0-22           10=96
        T3 & T1F        3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

First Class Mail *
        Use Copy Disk from Main Menu

FORMAT II *
        USE COPY DISK FROM MAIN MENU

FS-1 (FLIGHT SIMULATOR) (SUB LOGIC)
        T0                      10=96
        T1.5-T21 STEP 1.5       E=DB, F=AB, 10=BF, A=3, 4E=1
        T7-T8
        T9.5

GALACTIC GLADIATORS *
        T0-T20          10=B7, E=D7, 9=0, 31=0
        T21-T22         34=1

GORGON                  (SIRIUS)
        T0              10=96, 9=0
        T1.5-E.5        D=1, 24=96, A=3, E=DD, F=AD,
                        10=DA, 3B=40

HYPERSPACE WARS *       (CONTINENTAL)
        T0-T22          9=0

JAW BREAKER *           (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

KRELL LOGO *
        T0-T22

LIST HANDLER AND UTILITY *
        T1-T11
        T0              9=0, A=3, 44=1, 45=D, 50=3
        T12-T22.5 STEP .5       D=1, E=F5, F=D7, 10=F7
                                45=8, 46=D, 51=1
        (SEE NOTES FOR CHOPLIFTER)
```

```
MAGIC WINDOW *
        T0-T22

MICRO WAVE *            (CAVALIER)
        T0-T22
        T11             3B=1, A=1, 4B=1, 4D=8, 50=1

MOUSKATTACK *           (SIERRA ON-LINE)
        T0-T22          10=96
        SECTOR EDIT DOS 3.3 PATCHED
          TRACK 18, SECTOR 3
          CHANGE ADDRESS B1 FROM 49 TO 60

MULTI PLAN              (MICROSOFT)
        T0-T22          10=96

OLYMPIC DECATHALON *    (MICROSOFT)
        T0-T22          9=0

ORBITRON *
        T0-T1           9=0, 31=0
        T1.5-TF.5
        WRITE PROTECT COPY!

PFS & PFS REPORT        (SOFTWARE PUBLISHING CORP.)
        USE "COPY DISK" FROM MAIN MENU. AFTER COPYING AND BEFORE
        USING, PUT A TAB OVER THE WRITE PROTECT NOTCH OR  THE
        COPY WILL NOT WORK.

PHANTOMS FIVE           (SIRIUS)
        T0              9=0
        T2-T1C          3A=0, 50=20

PRISM *
        T0-T22

PRISONER *
        T0-T22

RASTER BLASTER          (OLD & NEW VERSIONS - BUDGECO)
        T0              10=96
        T5-T11  STEP 4  D=1, 9=0, 31=0, A=2, E=AD,
                        F=DE, 3B=40
        T6-T12  STEP 4
        T7.5-TF.5  STEP 4
        T1.5-T3.5  STEP 2

SABATOGE *
        T0-T22
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

SARGON *                (HAYDEN)
        T0-T1A

SCREENWRITER II *
        COPY DISK, THEN SECTOR EDIT
          DOS 3.3 PATCHED
          TRACK 3, SECTOR B
          CHANGE ADDRESSES 94, 95, 96 TO EA EA EA

SNOGGLE *               (BRODERBUND)
        T0-T9           9=0, 8=1

SPACE INVADERS *
        T0-T22          10=96

SNACK ATTACK            (DATA MOST)
        T0-T12
        SECTOR EDIT DOS 3.2 PATCHED
          TRACK 0 SECTOR 3
          CHANGE ADDRESS 63 FROM 38 TO 18

SNEAKERS                (SIRIUS)
        T0              9=0, 10=96, 44=1, 45=10, D=1
        T1.5-TC.5       44=0
        TD.5            44=1

SOFTPORN ADVENTURE      (ON-LINE)
        T0-T22          9=0
        T3              3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

SPACE EGGS *            (SIRIUS)
        T0              9=0
        T2-T6
        T11-1A

SPACE VIKINGS *
        T0-T22

SPEED READING *
        T0-T22          9=0, 10=96

SPIDER RAID *           (INSOFT)
        T0
        T1-T17          A=3, E=92, F=93, 4F=1, 10=95, 44=1
                        46=A, 9=0, 8=1, D=1, 24=96
                        3F=1, 34=1, 36=2A, 37=97
                        31=0, 43=0
        T1.5-T17.5      E=95, 10=92
        (SEE NOTES FOR CHOPLIFTER)
        (ONLY WORKS ON NEW VERSIONS)

STARBLASTER *
        T0              10=96, 9=0
        T7-T20 STEP 1.5 E=DF, F=AD, 10=DE

STARBLAZER              (BRODERBUND)
        SAME AS CHOPLIFTER

STARCROSS *             (INFOCOM)
        T0-T22          10=96

STELLAR INVADERS *      (APPLE)
        T0-T22
```

```
STOCK PORTFOLIO SYSTEM *
        T3-T22
        T0-T2           4=FD, 8=1, 10=AD

TAX MANAGER *           (MICROLAB)
        USE COPY DISK FROM MAIN MENU

TAX PREPARER *          (HOWARDSOFT)
        USE COPY DISK FROM MAIN MENU

THRESHOLD               (ON-LINE)
        T0-T22
        T1-T23 STEP 22  3B=1, A=1, 4B=1, 4D=8, 50=1
                        (ERROR 6 OK)

TUBE WAY *
        T0-T22

TYPING TUTOR *          (MICROSOFT)
        USE COPY DISK FROM MAIN MENU

ULTIMA II *
        COPY DISK, THEN SECTOR EDIT
          TRACK 3, SECTOR 0C
          CHANGE ADDRESSES 84, 85, 86 ALL TO EA.

ULTIMA II *
        T0-T22          10=96, 9=0, 34=1, 31=0

VERSAFORM *
        T0-T22

VISICALC                (VISICORP)
        T0-T16

VISICALC ///            (APPLE COMPUTER)
        T0-T22          10=96, 24=96, D=1

VISIDEX, VISISCHEDULE, VISITERM, VISITREND/VISIPLOT  (VISICORP)
        DON'T USE BIT COPY.  USE "COPY DISK" FROM MAIN MENU.

VISIFILE                (VISICORP)
        T0-T22          10=96, 34=1, 36=2A, 37=EB, 3E=2

WIZARDRY * (FRONT SIDE)
        COPY DISK THEN USE BIT COPY:
        T3-T23          10=96, 24=96, D=1
        NOTE: WRITE PROTECT BACKUP OR IT WILL NOT WORK.

WIZARDRY * (BACK SIDE)
        T1-T22          10=96, 24=96, D=1

WORD HANDLER *
        USE COPY DISK FROM MAIN MENU

ZARGS *                 (INOSFT)
        SAME AS SPIDER RAID
```

```
---------------------------------------

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

STARTING IN 86,SEARS WILL ISSUE THE
"DISCOVER" CREDIT CARDS.THESE SHOULD
SOON PROVE BETTER THAN VISA AND MASTER
CARD WITH CONSUMER AND DEALER DISCOUNTS
AND 35 BILLION IN CREDIT AVAILABLE.
 REMEMBER THE CHAOS WHEN VISA MAILED
OUT MILLIONS OF CREDIT CARDS! WATCH
YOUR MAILBOX (AND YOUR NEIGHBORS) IN
86 FOR YOUR VERY OWN UNUSED "DISCOVER"
CREDIT CARD(S).

NUFF SAID-
BOOTLEG
        ======================
        DOS Notes & Pointers..
        ======================

        -=> By THE FREEZE <=-

Suggested reading: Beneath Apple DOS
                   The DOS Manual

Needed equipment: Apple, Drive
                  IQ > 60

  I will start this article by giving
you an overview of what is in DOS and
what it does. First of all there is
the RWTS. This allows you to read or
write a sector at a time. All oper-
ations are done either directly or
indirectly through this. Starting at
$B600 and ending at $BFFF RWTS takes
up about 2.5K. Next is the File
Manager. This goes from $AAC9 to $B5FF
This is a bunch of subroutines which
execute your commands from basic.
Then there are the main DOS routines.
These interpret your commands and tell
the file manager what to do, which in
turn uses the RWTS to do them. These
routines go from $9D00 to $AAC8.
When you have MAXFILES set to 3, DOS
reserves memory from $9600 to $9CFF.
Setting MAXFILES higher will take up
more memory, lower than $9600.
There is another part of DOS, which
resides in the latter part of page 3
or from $3D0 to $3FF. This is called
the Dos Vector Table. I will go into
detail on that later.

 Well now, lets say you put a disk in
```

the drive and turned your computer on.
Then you loaded a file, edited it and
saved it. Why don't we take a look and
see exactly what is happening.

When you turn your computer on (if
you have autostart) the code on your
drive controller prom takes over.
This loads in a routine at $800.
This is called Boot 0. Then it jumps
to $801 and executes that code (boot
1). That code loads in sectors 1
through 9 which in turn loads in the
rest of DOS. Then it looks to see if
you have a HELLO program and jumps
to it. The first thing it does when
loading in a program, in this case
the HELLO program, is look at the
catalog track. Then after it finds
the file and the track and sector it
starts on, it reads in the first
sector. The first sector of a program
is called the Track Sector List or
TSL. This is a listing of all tracks
and sectors that have data for that
program. DOS reads this into memory
and then starts loading the program
in. But where does it know where to
load the program in and how does it
know what file type it is?
The file type was back on the catalog,
more (lots more) on that later...
On the first sector of data, not the
TSL, in the first two bytes is the
address to start loading in at. These
bytes as usual are in reverse order.
Well, now you know a little of how
DOS works. Lets go into more detail.

Here is where I will probably lose
you. If it gets confusing hang on.
Now we will look at track $11, which
is the catalog track. The VTOC or
Volume Table Of Contents is stored
at track $11, sector $00. This tells
DOS such things as: what sectors are
free, volume #, DOS version, first
link to catalog sector...
Bytes $01 & $02 of the VTOC tells us
where to find the
first catalog sector. This usually is
track $11, sector $0F. Byte $02 is
the DOS version. Either a "1", "2",
"3", for DOS 3.1, 3.2, 3.3, consec-
utively. Byte $07 is the volume #
usually $FE (254). The next thing of
interest is the Bit Map. Starting at
byte $38 you will see "FFFF0000".

For now, ignore the last two bytes.
The "FFFF" is a binary representation
of what sectors are free on a certain
track. In the two bytes there are
16 bits. Makes sense doesn't it, 16
bits and 16 sectors. If the bit is
set or a "1" then that sector is free.
If it is a "0" then it's used.
Now lets look at the catalog link.
On track $11, sector $0F, byte $01,
are two bytes that tell what track
and sector to find the first catalog
sector. This is almost always track 11
sector $0F. On track $11, sector $0F,
bytes 1 and 2, is a pointer to the
next sector, track $11, sector $0E.
The links continue until sector $01
where you will see zero's in those
bytes. I have been asked many times
how to get more than 105 files onto
a disk. If you edit the second and
third bytes on track $11, sector $01
to "100F", you will be able to use
track $10, sector $0F for a catalog
sector. You can continue on track
10 sector F and make a link to the
next sector and so on. Be sure to
mark it on the bit map or it will
get wiped out when DOS has to write
there. Well, we have covered most
of the VTOC, lets look at how the
catalog sectors are formatted.

Starting at byte $0B on any catalog
sector, is the entry for a file.
The first two bytes after that, tells
what track and sector the program
starts on. Then is the file type (more
on that later). Next comes the file
name, up to 30 characters. The last
byte before the next entry tells us
how many sectors the file takes.
This usually never goes over 255
sectors, however text files can take
more than 255 sectors. Now we can
look at the file type. We have to look
at this at the binary level. If the
first bit is set, it is a text file.
If the second is set, it's Integer.
Third is applesoft, fourth is binary.
If the eigth or MSB is set, the file
is locked. It's really quite simple.
$00 means a text file. $80 means a
locked text file. If it is a $84,
we have a locked binary file.

Now for the complicated stuff, how
DOS writes sectors, INITs a disk,

the "6 & 2" split. Lets say you put a blank disk in the drive, initialized it, and saved a file onto it. Lets see what happens. First off, at $A54F is the INIT routine. If you did A54FG from monitor, it would INIT your disk without a hello program. This lets your disk boot faster because it does not have to load in that file.
Ok. So you type in "INIT HELLO". DOS takes over and starts formatting your disk starting with track 0 and ending with track $23. Then it writes the catalog track and VTOC. Last it writes in DOS. Lets take a close look at a disk at the track level. First off we have what is called a GAP. This is made up of "FF"'s. Then we have the prolouge marks, ye olde "D5 AA 96". After that comes the volume, track, sector, check-sum, epilouge "DE AA EB". Then comes a smaller GAP with a different prolouge "D5 AA AD". Then $342 bytes of user data. Oops! $342 bytes of user data? I thought there were only $FF or 255 bytes per sector! (more on this later). Then we have the checksum. And last we have the epilouge "DE AA EB".
There are certain bytes that DOS doesn't write as data. These bytes are used in proulouge and epilouge marks. DOS looks for these when trying to find a sector. Now for the "6 & 2" split. The hardware on the apple doesn't allow for more than $3F different bytes to be written. That's why they used the "5 & 3" split on 3.2 disks. What that means is that from one byte, five bits are taken out and form one byte. The other three bits form one byte also. The six and two split is the same thing as the five and three but allows for more combinations.

Now for a little on copy protection. Back in the good 'ol days we could just demuffin everything. All they had was a modified DOS or changed address marks etc. After that they got a little smarter and some wrote their own DOS or used a modified RWTS. But nothing stops us pirates, all you had to do is read in data through their RWTS and write it back out standard. Then they got dirty, using the text page and the input buffer for data or code. They even used the stack (page 1) for code. To get around this, NMI card like crackshot and cracking chips were made. these dumped all memory to disk allowing the text page and the input buffer to be undisturbed. The newest thing seems to be SPIRAL TRACKING. The first game I saw this on was Maze Craze. Cracking it was quite easy though. All you had to do is cut out one part of disk access (at $855) that wasn't even needed. But who knows what we will be up against in the future.

I suggest you read "Beneath Apple DOS" and look at the DOS manual supplied with your Apple. This is for beginners or people who are too lazy to read a book...

I would appreciate lots of questions, I may not have made myself too clear or you may want to know more about a certain area. Just leave me e-mail.

The Freeze

D O S   T R I C K S

TRY THIS TO SEE ANY DOS, REMOVE THE REAR MOST SET OF RAM CHIPS FROM YOUR APPLE (THE ONES NEAR THE I/O SLOTS). THEN INIT A DISK, REPLACE THE RAM AND BOOT UP UNDER THE PROGRAM YOU WISH TO DEPROTECT. THEN FORCE A REBOOT WITH THE DISK YOU INITED IN DRIVE 1. THE DOS FROM THE PROTECTED DISK WILL (IN MOST CASES) STILL BE IN THE RAM UP TOP....

THIS NEW DOS IS A SLAVE AT 32 K AND THE OLD (AND PROTECTED DOS) IS STILL AT 48 K. THIS WILL WORK ON ABOUT 50% OF THE PROGRAMS. ENJOY

YOU CAN ALSO REMOVE THE TOP 32K AND GET TWICE AS MUCH.

=======================================

CHECKSUM TRICK

A VERY HANDY TECHNIQUE FOR TAKING A LOOK AT THE DATA ON A PROTECTED DISK IS TO DISABLE THE CHECKSUM IN THE RWTS. THE FORMATS OF MANY PROTECTED DISKS VARY ONLY IN THIS CHECKSUM, SO TURNING IT OFF SHOULD ALLOW ANY STANDARD TRACK/SECTOR UTILITY TO LOOK AT THE DISK! TO DO THIS, BOOT UP THE DOS THAT YOU WISH TO USE, AND ENTER THE MONITOR.

THEN ENTER B942:18 FOR DOS 3.3 OR B963:18 FOR DOS 3.2. THIS CHANGES A SET CARRY INSTRUCTION TO A CLEAR CARRY INSTRUCTION. NOW RETURN TO DOS AND RUN YOUR EDITOR. IF THE DISK YOU ARE LOOKING AT IS PROTECTED WITH THIS SYSTEM, YOU SHOULD BE ABLE TO READ IT NOW. TO MAKE THIS CHANGE TO A DOS ON A DISK, THIS DATA IS CONTAINED IN TRACK 0 SECTOR 3, AT EITHER BYTE $42 OR BYTE $63, FOR DOS 3.3 OR 3.2, RESPECTIVELY.
GOOD LUCK.......
RANDY

=======================================

TO AVOID RE-LOADING THE LANGUAGE CARD
ON BOOTUP (A MAJOR IRRITATION)
CHANGE THE FOLLOWING :

IN A 48K SYSTEM, CHANGE $BFCC TO 00 AND
$BFCF TO 00 : THIS WILL PREVENT THE
LANGUAGE CARD FROM BEING WRITTEN TO.
(INITIALIZE A DISKETTE WITH THIS DO
TO MAKE IT BOOT UP IN THIS FASHION)

(IF YOU LOOK AT THE CODE, YOU CAN MAKE
THE SAME MODS IN A COPY OF A SYST
MASTER ON THE DISK ITSELF, SO A MASTER
CREATE WILL PUT THIS DOS ON A DISKETTE.
CHANGE THE CODE THAT SAYS LDA C081 WITH
LDA C000 -- THAT SSHOULD WORK FINE.

=======================================

GET INTO MONITOR FROM A NORMAL DISK.
TYPE: 400<A800.ABFFM

POOF THERE YOU HAVE ALL THE DOS COMANDS
NOTICE THAT ALL THE LETTERS IN THE
COMAND ARE FLASHING BUT THE LAST ONE
THAT IS TO TELL YOU WHERE THE COMAND
ENDS. NOW NOTICE WHERE THE INIT,LOAD,
BLOAD,SAVE,BSAVE,CATALOG, ETC...
THEN BOOT SOMETHING LIKE BRAIN SURGEON
OR SOMETHING THAT HAS SOMETHING LIKE
A NORMAL FORMAT THEN TYPE THAT LINE
AND THEN YOU CAN SEE IF THEY CHANGED
ANY OF THE COMANDS!

=======================================
E.D.D. PARMAMETERS

A2-FS1:
   T0 - T6   INC 1.5

```
   T7 - T8
   T9.5-TA.5
   TC - T21  INC 1.5
ABM:NORM
A.E SIDE A:
   T1.5-TD.5
   TE -T18.5 INC 1.5
    SIDE B:NORM
ASCII EXPRESS PROFESSIONAL:NORM
ADVENTURE:NORM
AIRSIM-1:NORM
 WRITE-PROTECT BEFORE BOOTING!
ALGEBRA 1:NORM
ALKEMSTONE:NORM
APPLE PRESENTS- ERNIE'S QUIZ:NORM
APPLE PRESENTS- INSTANT ZOO:NORM
APPLE PRESENTS- SPOTLIGHT:NORM
APPLE PRESENTS- MIX AND MATCH:NORM
APPLE WORLD:
   T0-T23
APPLE WRITER:NORM
APPLE WRITER II:NORM
APPLE WRITER IIE:NORM
APPLE WRITER 80 COLMN PRE-BOOT:NORM
APVENTURE TO ATLANTIS:NORM
ARCADE MACHINE:
   T0 -T11
   T12.25-T21.25
ASTEROID FIELD:NORM
AUDEX:NORM
AZTEC:NORM
BANK STREET WRITER:
   T0 -T1A
   T1B-T22 PPM#3 OR #4
BATTLE FOR NORMANDY:SEE MINER 2049ER
BEER RUN:
   T0 PARM 28=2 OR 3
   T1.5-TD.5  PPM#2
BENEATH APPLE MANOR:(SPECIAL EDITION)
   T0-T22  PARM 0=3
BILL BUDGE 3-D GRAPHICS:NORM
BILL BUDGE SPACE ALBUM:NORM
BILL BUDGE TRILOGY OF GAMES:NORM
BORG:
   T1.5-TB.5
   TD-TE
   T0 PARM 28=2 OR 3
BUG ATTACK:
   T0-T22
   T1.5 =PPM#2
   T22 =PPM#2
BUSINESS GRAPHICS:NORM
CAMPAIGN TRILOGY:NORM
CANNONBALL BLITZ:NORM
CANYON CLIMBER:NORM
CARTELS AND CUTTHROATS:NORM
CASTLE WOLFENSTEIN:NORM
CCA DATA MANAGEMENT:NORM
```

```
CHECKERS (ODESTA):NORM (T0-T6)
CHESS 7.0 (ODESTA):NORM
CHOPLIFTER:
 NOTE:SOMETIMES VERY HARD TO COPY
   T0-TB PARM 28=2  00=3
   TC.25-T21.25
   T22
COMPUTER AMBUSH:NORM
COMPUTER AMBUSH VER 2:NORM
COMPUTER BISMARK:NORM
CONGO:NORM
COPTS & ROBBERS:SEE EPOCH
COPY II PLUS:NORM
CRANSTON MANOR:
   T0-T22
   T18 PPM#3
CRIME WAVE:NORM (T0-T11)
CRISIS MOUNTAIN:NORM
CRITICAL MASS:
 SIDE A:
   T0-TA
   T22  PPM#3
 SIDE B:NORM
CROSSFIRE:
   T0-T22
   T1 PPM#3
CROSSWORD MAGIC (BOTH SIDES):
   T0-T22 PPM#2
CUSTOM MICRO SYSTEMS ASSEMBLER:
   T0-T23:NORM
D.B. MASTER AND UTILITIES:
   T0 - T5
   T6.5-T22.5
DARK CRYSTAL:NORM
DATA TREE:NORM
DEADLINE:NORM
DESKTOP PLAN II:NORM
DISK EDIT 2.0 (DISK EDITOR):
   T0
   T1.5 -T5.5
   T21.25-T22.25
DISK RECOVERY:NORM
 IF THAT DOESN'T WORK, TRY:
   T0
   T1.25-T10.25  PPM#2
DOS ENHANCER:NORM
DUNG BEETLES:NORM
EASY-WRITER:NORM
EDU-PAINT:NORM
EINSTEIN COMPILER:NORM
ELECTRIC DUET:NORM
EMPIRE I: WORLD BUILDERS:NORM
EPOCH:
   T0 PARM 28=2 OR 3
   T1.5-TF.5 PPM#2
EVOLUTION:
   T0.25-T18.25
E-Z DRAW:NORM
```

```
FINANCIAL MANAGMENT SYSTEM III:
   T0-22
   T3 PARM 4=10 9=3 A=14 B=13 11=3;
      PPM#3 OR #4
FIRE BUG:NORM
GALACTIC EMPIRE:NORM
GALACTIC REVOLUTION:NORM
GAMMA GOBLINS:SEE BEER RUN
GAME SHOW & SUBJECTS:NORM
GERMAN/ENGLISH HANGMAN:NORM
GERMANY 1985:NORM
GORGON:
   T0  PARM 28=2 OR 3
   T1.5-TE.5  PPM#2
HADRON:SEE GORGON
HAIL:NORM
HEAD-ON:NORM
HELLFIRE WARRIOR:NORM
HOME ACCOUNTANT:NORM
INFORMATION MASTER:NORM
JAWBREAKER:
   T0-T22
   T3 PPM#3
KNIGHT OF DIAMONDS:PPM#2
L.A. LAND MONOPOLY:NORM
LABYRINTH:SEE CHOPLIFTER
LETTER PERFECT:NORM
LINGUIST:NORM
LIST HANDLER & UTILITIES:
   T11
   T12.25-T22.25  PARM 0=3
   T0  PARM 0=0 28=2
MASTER TYPE:NORM
MATH GAMES:NORM
MERLIN ASSEMBLER:NORM
MICROBE:NORM
MIDNIGHT MAGIC:
   T0 - T12
   T13.25-T15.25
   T22
MINER 2049ER:
   T1-T22
   T0  PPM#3 OR #4
MINGS CHALLENGE:SEE MINER 2049ER
MISSION ASTEROID:NORM
MOPTOWN:
   T0-T22  PARM 28=3
MILTIPLAN:
   T0-T22
   TA PPM#3 OR #4
MUSICOMP:NORM
NIBBLES AWAY II:NORM
NIGHT MISSION PINBALL:NORM
ODYSSEY:NORM
OLYMPIC DECATHALON:NORM
OLYMPIC INSURANCE SYSTEMS:NORM
PEGASUS II:SEE JAWBREAKER
PFM:NORM
```

```
PFS-FILE:
   T1-T22
   T0  PPM#3 OR #4
 >> WRITE-PROTECT BEFORE BOOTING !!! <<
PFS-FILE IIE:SEE PFS-FILE
PFS-GRAPH:SEE PFS-FILE
PFS-REPORT:SEE PFS FILE
PHANTOMS FIVE:SEE EPOCH
PINBALL CONSTRUCTION SET:NORM
POOL 1.5:PPM#2
PRESIDENT ELECT:NORM
PRISONER:NORM
PULSAR II:
   T: - T19
   T1A.5-T1D.5
QUEEN OF PHOBOS:NORM (T0-T1A)
REAR GUARD:NORM
RENDEZVOUS:SEE MINER 2049ER
RESCUE AT RIGEL:NORM
ROBOTWAR:NORM
SABATOGE:SEE JAWBREAKER
SARGON II:NORM
SCREENWRITER II:NORM
SEA FOX:SEE CHOPLIFTER
SENSIBLE SPELLER:NORM
SERIES RU-2:NORM
SERIES SP-2:NORM
SERIES FR-2:NORM
SERPENTINE:SEE CHOPLIFTER
SNEAKERS:SEE BEER RUN
SOFTPORN ADVENTURE:SEE JAWBREAKER
SORCEROR OF SIVA:NORM
SPACE EGGS:NORM
SPANISH/ENGLISH HANGMAN:NORM
SPECTRE:NORM
SPITFIRE SIMULATOR:NORM
SPY'S DEMISE:NORM
STARCROSS:NORM
STAR THIEF:
   T0-T13
   T22  PPM#3
SWASHBUCKLER:PARM 28=10
SUPER DISK COPY III:NORM
TWALA'S LAST REDOUT:PPM #2
TAXMAN:NORM
TEMPLE OF APSHAI:NORM
TERRORIST:NORM
 IF THAT DOESN'T WORK, TRY:
   T0-T1F
   T20.75-T22.75
THE ROUTINE MACHINE:NORM
THIEF:
   T0-T22
   T4-T5  PPM#2
THREE MILE ISLAND:NORM
THRESHHOLD:SEE CROSSFIRE
THUNDER BOMB:NORM (T0-T11)
TIC TAC SHOW:
   T0
   T1.5-T4.5
   T6-T22
  SERIES DISKS:NORM
TIME ZONE SIDE A:SEE MINER 2049ER
          SIDES B-L:NORM
TORPEDO FIRE:NORM
TRACK ATTACK:SEE CHOPLIFTER
TRANSEND: PPM#2
TRANSYLVANIA:NORM
TUBEWAY:NORM
TYPING TUTOR:NORM
ULTIMA:NORM
ULTIMA II:SEE RENDEZVOUS
ULYSSES:NORM
VISICALC 3.3:NORM
VISICALC 80 COLMN PRE-BOOT:NORM
VISICALC IIE:NORM
VISIDEX:NORM
VISIFILE:NORM
VISIPLOT:NORM
VISISCHEDULE:NORM
VISITERM:NORM
VOCABULARY BUILDER-FRENCH:NORM
VOCABULARY BUILDER-GERMAN:NORM
VOCABULARY BUILDER-SPANISH:NORM
WARP FACTOR:PPM#2
WIZARDRY:PPM#2
WORD HANDLER:
 NOTE: SOMETIMES VERY HARD TO COPY
   T11
   TB.25-T10.25
   T0-TA PARM 0=0 28=2
WORD RACE:PPM#2
WORLDS GREATEST BLACK-JACK:NORM
ZENITH:SEE CHOPLIFTER
ZOOM GRAPHICS:NORM
ZORK I:NORM
ZORK II:NORM
ZORK III:NORM
----------------------------------------
```

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

CBS IN NYC HAS AN EXPERIMENTAL RESEARCH
STATION GOIN AT 149.195,149.220 AND
149.245 MHZ TO DETERMINE THE FEASIBILIT
Y OF MOBILE SATELITE USE.

OTHER RESEARCH STATION ARE-

MOTOROLA-SCHAUMBERG,ILL. 1359.6 AND
1360.06 MHZ.

MOTOROLA-CANTON,MASS. SAME AS ABOVE

MOTOROLA-CUPERTINO,CA. SAME AS ABOVE

NUFF SAID-
BOOTLEG
       ATTENTION][E hackers! Having problems doing serious disk snooping because you can't reset to the monitor? For just 19.95 + 2.00 s/h I will send you the chip to make it all possible. Easily installed in 5 minutes with no cutting or soldering. These chips come programmed, tested, and ready to go. Cracking hints, tips and docs. included. Send check or money order to: Hacker Chips Inc. P.O. Box 2571 Hag. Md. 21740-2571. Allow 2 to 3 weeks for delivery. NOTE not available for the E or C YET!

MSG LEFT BY: SALLY RIDE
DATE POSTED:

HERE IS THE NEWEST FILE FROM THE LEGION
OF DOOM:
HACKING THE COSMOS PART 1
HERE IS A BRIEF DESCRIIPTION OF THE
MOST COMMONLY USED TRANSACTION CODES:
CAY-CREATE AN ASSEMBLY
DAY-DELETE AN ASSEMBLY
DRE-DENY AND RESTORE ESTABLISHMENT
FLR-FFRAME LAYOUT REPORT
ISH-INQUIRE ABOUT A CIRCUIT(PHONE #)
LO-LIST ORIGINATING LINE EQUIPMENT
MAL-MANUAL ASSIGNMENT LIST
MAY-MODIFY AN ASSEMBLY
MCH-MANUALLY CHANGE HUNT
MDC-MANUALLY DISCONNECT A CIRCUIT
SCA-SERVICE ORDER COMPLETION
SIR-SORTING INQUIRY BY RANGE
SLC-SUBSCRIBER LINE COUNTS FOR
    CUSTOM CALLING FEATURES
US- LIST USOC FILE DATA
WC- WIRE CENTER CHANGE
THOSE WILL BE DISCUSSED IN FURTHER
DETAIL IN PART 2.

PREFIXES, FORMATS AND CODE VALUES:
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
COSMOS PROVIDES A LANGUAGE BY MEAN OF
WHICH A USER CAN COMMUNICATE WITH THE
SYSTEM. THE LANGUAGE INCLUDES VARIOUS
PREFIXES AS WELL AS INPUT FORMATS AND

MSG LEFT BY: SALLY RIDE
DATE POSTED:

INPUT VALUES. PREFIXES ARE ABBREVIATION
WHICH REPRESENT SPECIFIC DATA CATAGORIE
TO THE SYSTEM WHEN INPUT BY THE USER.
AN EXAMPLE OF A PREFIX IS "TN"WHICH
MEANS "TELEPHONE NUMBER". AN INPUT
FORMAT DETERMINES THE NUMBER OF CHARAC-
TERS FOLLOWING A PREFIX AS WELL AS THE
PATTERN IN WHICH THESE MUST BE ENTERED.
FOR EXAMPLE, "TN XXX-XXXX" MEANS THAT
THE PREFIX "TN" MUST BE FOLLOWED BY

SEVEN CHARACTERS IN THE FORMAT SHOWN.

INPUT VALUES ARE THE ALLOWABLE DATA
ENTERED FOR EACH PREFIX IN THE CORRECT
INPUT FORMAT. AS MENTIONED IN THE
PREVIOUS PARAGRAPH THE INPUT FORMAT
FOR THE PREFIX "TN" IS "TN XXX-XXXX"
THE FIRST THREE CHARACTERS (XXX) MUST
BE ALPHANUMERIC; THE LAST FOUR (XXXX)
MUST BE NUMERIC. SO, COSMOS WOULD CON-
SIDER AN INPUT OF "TN 935-2481" AS
VALID INPUT. BUT YOU *MUST* USE THE
CORRECT WIRE CENTER FOR THE (XXX) IN
QUESTION. IN HACKING COSMOS PART 2
LEX WILL HAVE A LIST OF THE MOST
COMMONLY USED PREFIXES, FORMATS AND
PREFIX CODE VALUES WHICH ENABLE YOU
TO READ AND UNDERSTAND COSMOS TRANS-
ACTIONS.
SALLY RIDE:::SPACE CADET

MSG LEFT BY: SALLY RIDE
DATE POSTED:

COSNIX IS THE MUTATED VERSION OF COSMOS
AND UNIX BOTH WRITTEN BY BELL LABS.
COSNIX IS THE OPERATING SYSTEM OF THE
COSMOS SYSTEM.
SYSTEM COMMANDS------AS SOME OF YOU
WILL NOTICE, IF YOU READ THE BASICS OF
HACKING II, BY THE KNIGHTS OF SHADOW,
ALOT OF THE COMMANDS USED ON UNIX ARE
ALSO USED ON COSMOS. COMMANDS ARE AS
FOLLOWS::

```
WHERE---GIVES LOCATION OF THE SYSTEM::
        THIS COMMAND CAN BE VERY USEFUL
        SINCE YOU CAN GO TRASHING AT
        THE LOCATION THAT THE CENTER IS
        AT.
WC%WHERE====COSMOS 5
            STREET ADDRESS
            CITY, STATE  ZIP
WHAT----TELLS WHAT VERSION OF COSNIX
        THE SYSTEM IS RUNNING ON.
WC%WHAT==COSNIX OPERATING SYSTEM9.2.3
         RELEASE DECEMBER 7, 19831.2.2
         ETC.
JUST LIKE ON UNIX, TO SEE WHO ELSE IS
ON THE SYSTEM TYPE: WC%WHO
 COM3    TTOO  GB
 FW6     TTO4  HH, ETC.
```

COLUMN ONE BEING THE USERNAME, NEXT THE
TT#, AND LAST IS THE WIRE CENTER. SEE
THE CONTINUED CONCLUSION NEXT POSTING.

MSG LEFT BY: SALLY RIDE
DATE POSTED:

```
TO SEE WHAT YOU HAVE ACCESS TO TYPE:
WCLLS, OR WC%LS /* TO SEE ALL THE FILES
YOU HAVE ACCESS TO.
USE CAT/FILE-NAME TO SEE SOME MORE INFO
THIS WILL BE EXPLAINED IN MORE DETAIL
IN FUTURE EDITIONS.
DATE==SIMPLY GIVES THE DATE
USING CONTROL C WILL INTERUPT ANYTHING
YOU ARE EXECUTING AT THE TIME. YOU MAY
HAVE TO ENTER IT MORE THAN ONCE.

THAT IS ALL FOR PART 1, IT SHOULD GIVE
YOU A BASIC UNDERSTANDING OF COSMOS.
PART 2 WILL EXPLAIN AND SHOW YOU HOW TO
HOOK UP PHONE #'S AND SEE WHAT EQUIPMNT
IS ATTACHED, ALSO, WHAT THE ABBREVIA-
TIONS ARE SO YOU CAN UNDERSTAND IT ALL.

ACKNOWLEDGEMENTS: THE WARLOCK
                  BIOC AGENT 003
                  TUC-TUCBBS
WRITTEN BY: LEX LUTHOR

UPLOADED HERE BY:
     SALLY RIDE:::SPACE CADET
POST SOME DAMN MESSAGES YOU SLUGS!

APPLE CLONES

64K
CPM
NUMERIC KEYPAD
FUNCTION KEYS
AUTO REPEAT
100% APPLE COMPATABLE

$650

IBM CLONES
128K RAM
ROM CHIPS NOT INCLUDED
AVAILABLE IN PC OR XT VERSIONS

$650

CALL FOR DETAILS- 503-592-4461

NUFF SAID-
BOOTLEG

*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
       LOCKSMITH PARAMETERS LIST
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

     DONATED BY: SHERLOCK APPLE

ZORK (OLD VERSIONS)
     T0-T22: 1E=0B
     T3:    4C=1B (PATCH NC30 FOR VERSI
ON 4.0)
            4C=1B 57=00
E9=02  (USES NIBBLE COUNT SEE TECH NOTE
S) (VERSION 4.1 ONLY)
ZORK I AND ZORK II (NEW VERSIONS)
     T0-T22: 46=96 40=14

WARP FACTOR **     SAME AS TORPEDO FIRE
.FF4
WORD HANDLER
     T0: 46=96 54=12
-T22
     T1-TC: 44=FF 45=DF 46=DE (8 ERRORS
 O.K.)
.FF5
WORD HANDLER II
     T0: 46=96 54=12 53=00
     T11-T22
     T1-TC: 44=FF 45=DF 46=DE
     (NOTE-IF
AN 8 ERROR OCCURS RECOPY TRACK IT HAPPE
NED ON UNTIL GOOD.)

VISICALC (DOS 3.3 VERSION)
     T0-T15 NORMAL (T1 ERR IS OK)
VISICALC (APPLE ///)
     T0-T22 SYNC
VISIDEX (CHANGE AS OF 11-18-81)
     T0-T22: 40=04 16=08 41=FF 19=00 58
=0B 59=FF 81=AA 82=EB
83=FD 21=02
             46=96 54=12
VISIFILE     SAME AS DESK TOP PLAN II E
XCEPT PARM C0=FD SHOULD BE
C0=EC
VISISCHEDULE
     T0-T22: 40=04 16=08 41=FF 19=00 58
=0B 59=FF 81=AA 82=EB
83=EC 21=02 46=96 54=12
.FF3
VISITERM
     T0-T22 NORMAL
     T6: 40=08 16=08 41=FF 19=00 58=0B
59=FF 81=AA 82=EB 83=FC
.FF3
VISITREND/VISIPLOT
     T0-T22 NORMAL
     T7: 40=08 16=08 41=FF 19=00 81=DE
82=AA 58=0B 59=FF

U-BOAT COMMAND  **
     T0-T22: 4E=00 51=00 52=00 40=02 1E
=30 1B=19 1D=18 44=00
45=00 46=EB 47=AF
             48=FB 49=EB
.FF2
ULTIMA
```

```
     T0-T22: 1E=0B
ULYSIS **
     T0-T22 NORM
     T3: 4C=1B APPLY PATCH NC30 (VERSIO
N4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)

TAX PREPARER
     T0-T22: 46=96 54=12 4C=19
.FF4
THRESHOLD
     T0-T22 NORMAL
     T1-T23 BY 22: 4C=1B (PATCH NC30 FO
R VERSION 4.0)
                   4C=1B 57=00 E9=02  (
USES NIBBLE COUNT SEE TECH
NOTES) (VERSION 4.1 ONLY)
.FF2
TINY TROL
     T0-T22 NORMAL     T3.5-T5 BY 1.5
.FF2
TORPEDO FIRE
     T0 NORMAL     T1-T22: 4F=0B
.FF3
TWERPS **
     SAME AS GORGON
     PLUS T1C: 4C=1B 57=00 E9=02 D2=00
TWERPS  **
     T0: 18=20 19=00 46=96 4D=00 4E=00
52=00 53=00 54=12 57=00
40=20
     T1.5-TE.5 BY 1 SYNC: 72=00 73=00 7
7=00 78=00 79=12 7C=00
44=DD 45=AD 46=DA

SABATOGE **
     T0-T22 NORM
     T3: 4C=1B  APPLY PATCH NC30 (VERSI
ON 4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
SARGON II **
     T0-T1A NORM: 19=00 54=12 47=FF 4C=
18 48=FF 50=00 51=00 52=00
53=00
SCREENWRITER II **
     T0-T2: 4D=00
SHATTERED ALLIANCE
     T0-T22: 25=19
SHATTERED ALLIANCE (NEW)
     T0: 4C=18 47=FF 53=0B 54=12
     T1-T22: 44=D4 46=B7
.FF2
SINGA SHAPE MANAGER **
     T0-T22 SYNC
```

```
SNAKEBITE ** SAME AS GORGON
SNEAKERS
     T0: 18=20 19=00 46=96 4D=00 4E=00
52=00 53=00 54=12 57=00
40=20
     T1.5-TD.5 BY 1 SYNC: 72=00 73=00 7
7=00 78=00 79=12 7C=00
40=20 19=00 44=DD 45=AD 46=DA
.FF5
SNOGGLE **
     T0-T9 NORM
        OR
     T0-TF NORM     T10.5-T11.5 SYNC
.FF4
SOFTPORN ADVENTURE
     T0-T22 NORMAL (ALL VERSIONS)
     T3: 4C=1B APPLY PATCH NC30 (VERSIO
N 4.0 ONLY)
         4C=1B 57=00 E9=02  (USES NIBBL
E COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
.FF2
SOUTHERN COMMAND **
     T0-T22: 25=19 6B=00 34=D5 35=AB
.FF3
SPACE EGGS
     T0 NORM     T2-6 NORM     T11-13 N
ORM
     T14-1A: 44=DD
SPACE QUARKS
     T0: 18=50 19=00 40=20 46=96 4D=00
4E=00 52=00 53=00 54=12
57=00
     T1-T2: 44=AB 45=D4 46=AB
     T3.5-T5.5 BY 1     T7
     T9: 44=FE 45=DD 46=AF
     TA.5-B.5 BY 1: 44=AA 45=DE 46=BB
     TD-15 BY 1
SPACE WARRIOR
     T0: 18=50 19=00 40=20 46=96 40=20
4E=00 52=00 53=00 54=12
57=00
     T2.5-T3.5: 44=DF 45=AD 46=DE
     T5-T8 BY 3     T6.5     TA-T10 BY
3
STAR BLASTER  **
     T0 NORM
     T7-T20.5 BY 1.5 SYNC: 72=00 73=00
77=00 78=00 79=12 7C=00
40=20 19=00 44=DF 45=AD 46=DE
STAR CRUISER **
     T0-T3 BY 3 SYNC     T5-TB BY 1 SYN
C     T11-T12 BY 1 SYNC
     T4 SYNC: 44=AA 45=DD 46=BB
STAR MINES **
     T0 NORM
     T1-T2 NORM: 46=AD
     T4-TA NORM
```

```
STAR RATERS **
     T0-T5 NORM (TRACK 5 ERROR MAY OCCU
R)
STAR THIEF
     T0-T13 NORMAL (TRACK E-13 ERRORS M
AY OCCUR) (ALL VERSIONS)
     T22: 4C=1B APPLY PATCH NC30 (VERSI
ON 4.0 ONLY)
          4C=1B 57=00 E9=02  (USES NIBB
LE COUNT SEE TECH NOTES)
(VERSION 4.1 ONLY)
.FF2
SUPER APPLE BASIC **
     T0-T22 NORM     T3 NORM-EXTENDED R
ETRY
.FF3
SUPERSCRIBE II
     T0-T22 NORM
     T3 NORM: 45=00 50=00
SUPERSCRIBE II **     SAME AS PEGASUS I
I
.FF2

RASTER BLASTER (FOR OLD RASTER BLASTER
ONLY)
     T0 NORMAL
     T5-T11 BY 4 SYNC: 44=AD 45=DE 53=0
0
     T6-T12  BY 4 SYNC     T7.5-TF.5 BY
 4 SYNC     T1.5-T3.5 BY 2
SYNC
.FF4
RASTER BLASTER (NEW VERSIONS)
     T0: 46=96 54=12
     T5-T11 BY 4 SYNC: 44=AD 45=DE 46=0
0 72=00 73=00 75=00 78=00
79=12
     T6-T12 BY 4 SYNC     T7.5-TF.5 BY
4 SYNC     T1.5-T3.5 BY 2
SYNC
.FF3
RETROBALL  **
     T0, T4-T6, T
----------------------------------------

YEP-WEVE BEEN A LITTLE LATE WITH THIS
ISSUE DUE TO MOVING.
STILL HAVE THE SAME PHONE NUMBERS,
BUT OUR CURRENT ADDRESS TIS-

     THE BOOTLEGGER/HACKER MAGAZINE

         1080 HAYS CUT-OFF ROAD
         CAVE JUCTION,OR.97523
2
```

```
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

[[ PRESS SPACEBAR TO QUIT ]]

        NIBBLES AWAY  PARAMETERS

COMPANY NAME:
                    AUTO-LOAD FILE
PROGRAM NAME        COPY TRACKS    PARA
METERS TO CHANGE        TO USE
----------------    -----------    ----
----------------   ----------------
A P P L E   C O M P U T E R
SUPER PILOT ------- 0-0............ADDR
=D5 AA 96
                    2-22
                        SECTMOD [F=16,C
=OFF,T=0,S=0A]
                          CHANGE ADDRES
S 79 FROM 43 TO EA
                          CHANGE ADDRES
S 7A FROM 41 TO EA
                          CHANGE ADDRES
S 7B FROM C6 TO EA

A U T O M A T E D   S I M U L A T I O N
 S:
TEMPLE OF APSHAI -- 0-22...........ADDR
=D5 AA B5

A V A N T E - G A R D E
HI-RES SECRETS ---- 0-22...........ADDR
=D5 AA 96

B R O D E R B U N D   S O F T W A R E:
WARLORDS ---------- 0-F............ADDR
=D5 AA B5

C E N T R A L   P O I N T   S O F T W A
 R E:
COPY ][ PLUS ------ 0-2............NORM
AL
                                   DEL
BYTE =20
D A T A   M O S T:
SPACE KADET ------- 0-22...........ADDR
=D5 AA 96
MARS CARS                          OVER
IDE STANDARDIZER
CRAZY MAZEY
TAX BEATER -------- 0-22...........ADDR
=D5 AA 96
REAP                    SECTMOD [F=16,C
=OFF,T=0,S=03]
                          CHANGE ADDRES
S 42 FROM 38 TO 18
MONEY MUNCHER ----- 0-22...........ADDR
=D5 AA 96
```

```
E D U W A R E:
THE PRISONER ------ 0-22...........SYNC
ALGEBRA I --------- 0-22...........ADDR
=D5 AA B5
EMPIRE 1 WORLD ---- 0-22...........ADDR
=D5 AA 96
BUILDERS            3-3............NIBB
LE COUNT
PRISONER ][ ------- 0-22...........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=ON,T=1F,S=0E]
                         CHANGE ADDRESS
 D5 FROM AD TO 2F
                         CHANGE ADDRESS
 D6 FROM 99 TO AF
                         CHANGE ADDRESS
 D7 FROM F0 TO 32
I N F O C O M:
STARCROSS --------- 0-22...........ADDR
=D5 AA 96

I N S O F T:
ELECTRIC DUET ----- 0-22...........ADDR
=D5 AA 96
                                   INS=
 DE AA EB
                                   OVER
IDE STANDARDIZER
                                   FIX
AMNT=04
I N T ' L   S O F T W A R E   M K T G
MATH MAGIC -------- 0-22...........NORM
AL

I D S:
PRISM PRINT ------- 0-21...........ADDR
=D5 AA 96
                                   OVER
IDE STANDARDIZER
                        SECTMOD [F=16,C
=ON,T=21,S=00]
                          CHANGE ADDRES
S 27 FROM FB TO 22

L E A R N I N G   C O M P A N Y
BUMBLE GAMES ------ 0-22...........ADDR
=D5 AA 96
BUMBLE PLOT         NOTE: WRITE PROTECT
 BEFORE BOOTING!
ROCKY'S BOOTS
JUGGLER'S RAINBOW

M I C R O L A B
JIGSAW ------------ 0-0............NORM
AL
                    A-17...........NORM
```

```
AL
                    1-9...........ADDR
=D3 96 F2
M U S E:
BEST OF MUSE ------ 0-22..........SYNC
THREE MILE ISLAND
GLOBAL WAR

M I C R O S O F T:
OLYMPIC DECATHALON  0-22..........ADDR
=D5 AA B5

O N L I N E   S Y S T E M S:
GENERAL MANAGER --- 0-22..........ADDR
=D5 AA 96
V1.5                    SECTMOD [F=16,C
=ON,T=1F,S=0E]
                          CHANGE ADDRES
S C1 FROM -- TO 4B
                          CHANGE ADDRES
S C2 FROM -- TO E0
                          CHANGE ADDRES
S C3 FROM -- TO 49
                        SECTMOD [F=16,C
=ON,T=21,S=01]
                          CHANGE ADDRES
S 2E FROM -- TO 60
SABOTAGE ---------- 0-22..........NORM
AL
ALIEN RAIN
SNOGGLE ----------- 0-22..........ADDR
=D5 AA B5

TIME ZONE V1.1 ---- 0-22..........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=ON,T=03,S=0B]
                         CHANGE ADDRESS
 F0 FROM 20 TO EA
                         CHANGE ADDRESS
 F1 FROM 00 TO EA
                         CHANGE ADDRESS
 F2 FROM 17 TO EA

P E N G U I N   S O F T W A R E:
PIE MAN ----------- 0-22..........ADDR
=D5 AA 96

P H O E N I X   S O F T W A R E:
ZOOM GRAPHICS ----- 0-22 BY 2......ADDR
=D5 AA 96
2ND EDITION                        INS=
DD AA ED B5
                    1-21 BY 2......ADDR
=D4 AA 96
                    N O T E: WRITE PROT
ECT BEFORE BOOTING!!
```

```
ADVENTURE IN TIME - 0-C...........NORMA
L
BIRTH OF THE ------ 0-9...........NORMA
L
PHOENIX

P I C A D I L L Y   S O F T W A R E:
FALCONS ----------- 0-0............ADDR
=D5 AA B5
                    1.5-4.5X1.5....ADDR
 DF AD DE
                    5.5-5.5X1
                    7-AX1
                    B.5-E.5X1.5
                    10-12X1
                    13.5-14.5X1
                    16-19X1.5
                    1A-1B.5X1.5

S E N S I B L E   S O F T W A R E:
IMAGE PRINTER ----- 0-2............ADDR
=D5 AA 96
                    3-7............ADDR
=F7 AA 96
                    9-22
                     SECTMOD [F=16,C=OF
F,T=0,S=03]
                       CHANGE ADDRESS 4
2 FROM 38 TO 18
                     SECTMOD [F=16,C=OF
F,T=2,S=03]
                       CHANGE ADDRESS 2
A FROM 2C TO 4C
                       CHANGE ADDRESS 2
B FROM 06 TO 5D
                       CHANGE ADDRESS 2
C FROM B7 TO B4
SUPER DISK COPY --- 0-22...........ADDR
=D5 AA 96
(VERSION 3.7)                       ERR
ORS OK
THE BUG ----------- 0-0............NORM
AL
                    15-15..........GAP
BYTE 2=FF
                                   GAP
SIZE=10
                    16.5-16.5

S E R I U S   S O F T W A R E:
KABUL SPY --------- 0-21...........ADDR
=D5 AA 96
(BOTH SIDES)         SECTMOD [F=16,C=OF
F,T=0,S=0
                       CHANGE ADDRESS 4
9 FROM -- TO EA
                       CHANGE ADDRESS 4
A FROM -- TO EA
```

```
                       CHANGE ADDRESS 4
B FROM -- TO EA
DARK FOREST ------- 0-22...........ADDR
=D5 AA B5
                                   OVER
IDE GLITCH DETECT

S I L I C O N   V A L L E Y   S O F T W
 A R E:
WORD HANDLER ][ --- 0-0C...........ADDR
=FF DF DE
                    11-22..........ADDR
=D5 AA 96
S O F T A P E:
DRAW POKER -------- 0-22...........ADDR
=D5 AA B5

S O F T W A R E   P U B L I S H I N G
 C O R P.:
PFS/PFS REPORT ---- 0-13...........ADDR
=D5 AA 96
(REVISED)                          OVER
IDE STANDARDIZER
                                   GAP
BYTE 1=C0, GAP BYTE 2=D0
                                   FILT
ER=C0-C8 (NO INVERSE)
                    N O T E: WRITE PROT
ECT BEFORE BOOTING!!
PFS GRAPH --------- 0-22...........ADDR
=D5 AA 96
                                   OVER
IDE STANDARDIZER
                                   GAP
BYTE 1=C0, GAP BYTE 2=D0
                                   FILT
ER=C0-C8 (NO INVERSE)

S P E C I A L   D E L I V E R Y   S O F
 T W A R E:
UTOPIA GRAPHICS --- 0-22...........ADDR
=D5 AA 96
SYSTEM                             TURN
 ON 3.3 FILTER
                       SECTMOD [F=16,C=
ON,T=0,S=0]
                         CHANGE ADDRESS
 42 FROM 38 TO 18
GALACTIC WARS ----- 0-22...........ADDR
=D5 AA 96
BRIDGE TUTOR

S T O N E W A R E:
D B MASTER -------- 0-5............ADDR
=D5 AA 96, SYNC
UTILITY PAC #1      6.5-22.5.......SYNC

S T R A T E G I C   S I M U L A T I O N
```

```
 S:
BATTLE OF SHILOH -- 0-22...........ADDR
=D4 AA B7
WARP FACTOR

S Y T O N I C   S O F T W A R E:
INTERLUDE ----------0-22...........ADDR
=D5 AA B5

X P S:
APPLE CILLIN ------ 0-0............ADDR
=D5 AA 96
                    1-22...........ADDR
=D5 AA B5
                    11-11..........ADDR
=D5 AA 96

                PARAMETERS:  OCTOBER 19
82

COMPANY NAME:
PROGRAM NAME        COPY TRACKS    PARA
METERS TO CHANGE
----------------    -----------    ----
----------------
A D V E N T U R E   I N T E R N A T I O
 N A L:
ELIMINATOR -------- 0-21...........ADDR
=D5 AA 96
                        SECTMOD [F=16,C
=OFF,T=03,S=0D]
                          CHANGE ADDRES
S 2E FROM 20 TO EA
                          CHANGE ADDRES
S 2F FROM 30 TO EA
                          CHANGE ADDRES
S 30 FROM 72 TO EA

A P P L E   C O M P U T E R:
VISICALC /// ------ 0-22...........SYNC
APPLE WRITER /// -- 0-22...........SYNC
APPLE LOGO -------- 0-22...........ADDR
 D5 AA 96
                    1-1............ADDR
 AA D6 EE
                                  NIBBL
E COUNT=Y
                                     FI
ND MAX=03
                                   SHIF
T N+ = 08
                                   SHIF
T N- = 00

APPLE WRITER ][ --- 0-3............ADDR
 D5 AA DA (OR D5 AA DB)
                    4-22...........ADDR
 D5 AA 96
```

```
A V A N T E - G A R D E   C R E A T I O
 N S
ZERO GRAVITY PINBALL 0-22...........ADD
R=D5 AA B5

B P I:
   (REVISED)
ACCOUNTING -------- 0-22...........ADDR
=D5 AA 96
  SYSTEM                   FIX AMNT=04
,  GAPBYTE1=C8
                           GLOBAL MOD
BYTE D972 FROM 03 TO 00
                    11-11..........INS=
AD FB E6 FF E6
                                   SYNC
 SIZ=0A

B R O D E R B U N D   S O F T W A R E:
APPLE PANIC ------- 0-D
GENETIC DRIFT ----- 0-0............ADDR
=D5 AA B5
                    1-3............ADDR
=BB D5 BB
                    4.5-6 BY 1.5
                    7.5-B.5
                    D-D............ADDR
=D4 D5 BB
                    E.5-12.5.......ADDR
=AD B5 DE

SPACE QUARKS ------ 0-0............ADDR
=D5 AA B5
                    1-2............ADDR
=FF DF DE, DATA MAX=25
                    3.5-5.5
                    7-9 BY 2
                    A.5-B.5
                    D-15

SPACE WARRIOR ----- 0-0............ADDR
=D5 AA B5, DATA MAX=30
                    2.5-3.5........ADDR
=DF AD DE
                    5-8 BY 3
                    6.5-6.5
                    A-10 BY 3

B U D G C O:
RASTER BLASTER ---- 0-0............ADDR
=D5 AA 96, SYNC
                                   DATA
 MIN=18, DATA MAX=40
                    5-11 BY 4......ADDR
=AD DE, DATA MIN=13, SYNC
                    6-12 BY 4......SYNC
                    7.5-F.5 BY 4...SYNC
                    1.5-3.5 BY 2...SYNC

C A V A L I E R   C O M P U T E R:
MICROWAVE --------- 0-22...........ADDR
=D5 AA 96
                      SECTMOD [F=16,C=O
N,T=02,S=01]
                         CHANGE ADDRESS
 DA FROM A9 TO AD
                         CHANGE ADDRESS
 DB FROM 60 TO 03
                         CHANGE ADDRESS
 DC FROM 8D TO 81
                         CHANGE ADDRESS
 DD FROM 7E TO 60

C O N T I N E N T A L   S O F T W A R E
:
GUARDIAN ---------- 0-1............ADDR
=D5 AA B5
                    2-11...........ADDR
=D6 AA B5
                                   INS=
DF AA EB F7, SYNC SIZ=0A
D A T A   M O S T:
COUNTY FAIR ------- 0-22...........ADDR
=D5 AA B5
SNACK ATTACK         SECTMOD [F=13,C=OF
F,S=03,T=00]
                       CHANGE ADDRESS 6
3 FROM 38 TO 18
SNACK ATTACK ------ 0-22...........ADDR
=D5 AA B5
(REVISED)            SECTMOD [F=13,C=OF
F,S=01,T=00]
                       CHANGE ADDRESS 3
9 FROM 38 TO 18

SWASHBUCKLER ------ 0-22...........ADDR
=D5 AA 96
CASINO 21            SECTMOD [F=16,C=OF
F,S=03,T=00]
                       CHANGE ADDRESS 4
2 FROM 38 TO 18

D A T A   S O F T:
DUNG BEETLES ------ 0-0............ADDR
=D5 AA B5
                    1-1............ADDR
=F5 F6 F7
                    4-22
                     SECTMOD [F=13,C=ON
,T=00,S=01]
                         CHANGE ADDRESS
 6D FROM 01 TO 7B
                         CHANGE ADDRESS
 6E FROM 61 TO 69
```

```
G E B E L L I   S O F T W A R E:
FIREBIRD ---------- 0-0...........ADDR
=DD AD DA, SYNC
                    1.5-B.5.......SYNC

H O W A R D S O F T:
TAX PREPARER ------ 0-22..........ADDR
=D5 AA 96

I N F O C O M:
DEADLINE ---------- 0-22..........ADDR
=D5 AA 96

I N N O V A T I V E   D E S I G N   S O
 F T W A R E:
POOL 1.5 ---------- 0-15..........ADDR
=D5 AA B5
                    1E-21
                      SECTMOD[F=13,C=OF
F,T=0B,S=07]
                         CHANGE ADDRESS
 6A FROM 8D TO 60
```

STEP BY STEP GUIDE TO BACKING-UP DISKS
WITH
NIBBLES AWAY][

THERE ARE THREE BASIC STEPS TO BACKUP A DISKETTE:

1. LOCATE THE TRACKS WHICH CONTAIN DATA.
2. FIND THE ADDRESS MARKER FOR THE SECTORS THERE.
3. FIGURE OUT ANY ADDITIONAL PROTECTION.

(HINT: #3 IS THE HARD ONE!)

FOR MOST OF THE PROCEDURES BELOW, A BASIC WORKING KNOWLEDGE OF THE TRACK/BIT EDITOR (TBE) IS REQUIRED. FOR THOSE WHO ARE NOT FAMILIAR WITH THE TBE, AN OVERALL DESCRIPTION AND SOME EXAMPLES ARE GIVEN BELOW. THE EXAMPLES ARE EASIER TO UNDERSTAND IF THEY ARE PERFORMED AS YOU READ THIS, SO YOU MAY WANT TO BOOT UP NIBBLES AWAY][AND TRY THEM OUT TO GET A BETTER UNDERSTANDING OF WHAT IS GOING ON.

ENTER THE TBE BY SELECTING OPTION 'T' FROM THE MAIN MENU. A LARGE SECTION OF NUMBERS WILL APPEAR ON THE SCREEN, WITH TWO DASHED LINES AT THE TOP. THE INFORMATION IN BETWEEN THESE LINES IS THE STATUS INFORMATION AND INFORMS YOU OF SUCH THINGS AS CURSOR POSITION, TRACK NUMBER, AND IS ALSO THE LOCATION WHERE VARIOUS PROMPTS APPEAR FOR CERTAIN FUNCTIONS. THE NUMBERS AT THE BOTTOM ARE SEPARATED INTO TWO SECTIONS. ON THE LEFT ARE THE STARTING MEMORY ADDRESS'S FOR EACH LINE TO THE RIGHT. MOVE THE CURSOR AROUND USING I,J,K OR M, AND WATCH THE ADDR INDICATOR IN THE STATUS LINE. IT WILL TELL YOU EXACTLY WHAT MEMORY ADDRESS THE VALUE UNDER THE CURSOR REPRESENTS. THE ARROW KEYS CHANGE THE AREA OF MEMORY WHICH YOU CAN SEE. THEY SHIFT YOUR VIEW 256 BYTES FORWARD OR BACKWARD AT A TIME. THE ONLY REALLY IMPORTANT THING TO KNOW FOR THIS DISCUSSION IS HOW TO USE THE ARROW KEYS TO MOVE THE VIEWING 'WINDOW' AROUND IN MEMORY.
THE ';' (UNSHIFTED '+') AND THE '-' KEYS INCREMENT AND DECREMENT THE

TRACK NUMBER IN THE STATUS LINE. PRESSING 'R' WILL CAUSE DRIVE ONE TO READ THE DATA FROM THE TRACK INDICATED IN THE STATUS LINE INTO MEMORY. THE BYTES ON THE SCREEN WILL CHANGE, SINCE DIFFERENT DATA HAS BEEN READ IN. PRESSING THE 'R' KEY MULTIPLE TIMES WILL RESULT IN DIFFERENT DATA BEING DISPLAYED. THIS IS BECAUSE NIBBLES AWAY][STARTS READING AT WHATEVER POINT HAPPENS TO BE UNDER THE HEAD WHEN THE DRIVE IS TURNED ON, WHICH IS RANDOM, HENCE THE CHANGE IN THE DISPLAYED DATA (THE DATA IS NOT ACTUALLY DIFFERENT, IT IS JUST NOT LOADED AT THE SAME MEMORY LOCATION AS IT WAS PREVIOUSLY).

STEP 1:
TO DO THIS WE MUST LOCATE ALL OF THE TRACKS ON THE DISK WHICH CONTAIN DATA. TO DO THIS WE SHOULD HAVE THE TRACK POINTER SET TO TRACK 00. PRESSING 'R' WILL READ IN THE TRACK AND SHOW IT ON THE SCREEN. THE ARROW KEYS SHOULD BE USED TO MOVE THE VIEWING 'WINDOW' TO START AT $2000. NOW WE WILL MOVE FORWARD AND TRY TO DETERMINE IF THIS TRACK CONTAINS VALID DATA. ACTUALLY, TRACK 00 MUST CONTAIN SOME DATA IN ORDER FOR THE DISK TO BOOT, BUT WE WILL BE USING THIS PROCEDURE ON OTHER TRACKS WHICH DO NOT NECESSARILY CONTAIN DATA.

THE MAIN THING WHICH WILL IDENTIFY A TRACK AS CONTAINING DATA IS THE PRESENCE OF GAPS. GAPS ARE SECTIONS OF THE SAME BYTE REPEATED SEVERAL TIMES. NORMALLY THEY ARE MADE UP OF $FF'S AND ARE 6-20 IN LENGTH. TO SEE WHAT THESE LOOK LIKE, INSERT YOUR SYSTEM MASTER DISK AND READ IN TRACK 00 AS DESCRIBED ABOVE. MOVING THROUGH THE BUFFER WITH THE ARROW KEYS WILL REVEAL A LARGE VARIETY OF VALUES. SPACED OUT AMONG THESE SHOULD BE SECTIONS OF FF'S WHICH CONTAIN ABOUT 6-20 IN A ROW, DEPENDING ON THE EXACT DISK. NORMALLY DOS 3.2 DISKS HAVE LARGER GAPS THAN DOS 3.3 DISKS. THERE SHOULD BE MANY OCCURANCES OF THE GAPS, SPACED OUT SO THAT YOU SEE ONE ABOUT EVERY OTHER TIME THAT YOU USE THE ARROW KEYS TO MOVE FORWARD OR BACKWARD.

NOTE:YOU MAY SEE A SECOND, SMALLER (2-5 $FF'S), GAP FOLLOWING A LARGE GAP, WITH A SMALL SECTION OF DATA IN BETWEEN. THIS IS CALLED THE SECONDARY GAP. WHEN REFERING TO A GAP HERE, WE WILL ALWAYS BE TALKING ABOUT THE PRIMARY GAP, NOT THE SECONDARY ONE.

NOW TRY LOOKING AT OTHER TRACKS ON THE DISK. FIRST LOOK ONLY AT THE FULL TRACKS (NO .5 ON THE END). ALL OF THEM WILL BE SIMILAR TO TRACK 00 IN THE APPEARANCE OF THE GAPS. YOU MAY WANT TO TRY THIS SEVERAL TIMES TO BECOME COMFORTABLE WITH LOCATING GAPS ON A GIVEN TRACK.

NOW READ IN A HALF TRACK (.5 ON THE END). SCAN MEMORY TO LACATE SOME OF THE GAPS. SINCE SYSTEM MASTER DISKS DO NOT USE HALF-TRACKS, THE DATA WHICH WE SEE HERE IS REALLY 'CROSS-TALK'. IN OTHER WORDS, DATA WAS WRITTEN ON THE FULL TRACK, BUT THE MAGNETIC PATTERN SPREAD OUT A BIT, AND SO WE SEE SOME DATA HERE. THE TELL-TALE SIGN OF THIS PHENOMENA IS THAT THE GAPS WILL NOT BE ALL THE SAME. THAT IS, THEY MAY HAVE ONE OR MORE VALUES IN THEM WHICH ARE NOT CONSISTENT. THIS TELLS US THAT THERE IS SOME DATA ON THIS TRACK, BUT THAT IT IS NOT VALID DATA. TAKE A LOOK AT SOME OTHER HALF-TRACKS SO THAT YOU CAN TELL IF YOU ARE LOOKING AT A FULL TRACK OR A HALF TRACK BY EXAMINING THE GAPS.

THE NEXT ITEM WHICH YOU NEED TO BE ABLE TO IDENTIFY IS A BLANK TRACK. TO DO THIS, INSERT A BLANK (NON-INITIALIZED) DISK INTO DRIVE ONE. READ ANY TRACK ON THIS DISK AND SCAN THROUGH THE MEMORY ADDRESSES. THERE WILL BE NO GAPS FOUND, AND MANY OF THE BYTES SEEN ON A TRACK LIKE THIS WILL END IN 0 (I.E. $A0,$B0,$E0), WHICH ARE NOT LEGAL DISK BYTES. THIS MEANS THAT THE CONTROLLER CAN FIND NO VALID DATA ON THE TRACK. SOME DISKS HAVE PORTIONS OF TRACKS WHICH ARE NOT USED, SO YOU SHOULD ALWAYS BE SURE TO EXAMINE AT LEAST 24 SCREENFULLS OF INFORMATION TO MAKE SURE THAT THERE IS NO DATA AT ANY POINT ON THE TRACK.

OUR NEXT TOOL FOR FINDING DATA IS THE FACT THAT VALID DATA MUST BE AT LEAST 1 TRACK APART. IN OTHER WORDS, IF YOU LOCATE DATA ON TRACK 3.5, THEN TRACK 4 CANNOT HAVE DATA AND THE NEXT PLACE WHERE DATA CAN BE IS TRACK 4.5. THIS IS VERY HELPFUL FOR FINDING TRACKS WITH DATA.

NOTE: IF YOU LOCATE DATA ON A GIVEN TRACK, IT IS A GOOD IDEA TO LOOK AT THE TRACKS ONE HALF TRACK TO EITHER SIDE, TO MAKE SURE THAT THEY LOOK LESS VALID THAN THE ONE THAT YOU HAVE SELECTED AS THE REAL ONE.

WELL, NOW THAT WE KNOW HOW TO LOACATE DATA ON A TRACK, WE CAN BEGIN AT TRACK 0 AND STEP TOWARDS TRACK 22, CHECKING EACH TRACK TO SEE IF IT APPEARS TO HAVE DATA ON IT. MOST DISKS HAVE A PATTERN TO THE POSITION OF THE DATA, AND IF YOU CAN FIGURE IT OUT, YOU MAY BE ABLE TO JUST CHECK A FEW TRACKS TO MAKE SURE, AND THEN GO ON TO STEP 2. OTHERWISE THE DATA MUST BE LOCATED ONE TRACK AT A TIME.

MOST DISKS USE THE STANDARD TRACKS (1,2,3,...,22), BUT THERE ARE SOME WHICH USE HALF-TRACKS AND SOME WHICH USE ALL THE WAY OUT TO TRACK 23 (WHICH, BY THE WAY CANNOT BE READ ON ALL DRIVES SINCE NO DRIVES WERE EVER DESIGNED TO GO OUT THAT FAR).

WHEN ALL TRACKS WHICH CONTAIN SOME TYPE OF DATA ARE LOCATED, WE CAN MOVE ON TO STEP 2.

STEP 2:

NOW WE MUST TELL NIBBLES AWAY][HOW TO READ THE INFORMATION ON THE TRACKS WHICH WE HAVE FOUND TO CONTAIN VALID DATA. THIS IS DONE BY GOING BACK TO EACH OF THESE TRACKS WITH THE TBE AND FINDING THE ADDRESS MARK FOR EACH ONE. THE ADDRESS MARK WILL BE THE FIRST 3 BYTES FOLLOWING THE GAP. TO SEE THIS IN OPERATION, TAKE A LOOK AT A TRACK FROM YOUR SYSTEM MASTER DISK. AFTER EACH GAP YOU WILL SEE EITHER 'D5 AA 96' FOR A DOS 3.3 MASTER DISK, OR 'D5 AA B5' FOR A DOS 3.2 DISK. THESE VALUES SHOULD BE NOTED DOWN ALONGSIDE OF EACH TRACK NUMBER WHICH CONTAINS DATA. MANY TIMES THERE WILL BE ONLY ONE, OR MAYBE 2 PATTERNS FOR ALL TRACKS.

AFTER THIS, WE ARE READY TO BACK-UP THESE TRACKS. THIS IS DONE BY EXITING THE TBE (USE 'Q') AND THEN SELECTING 'M' FOR THE MODIFIERS MENU. THEN SELECT 'B' FOR BACKUP MODIFIER. WHEN ASKED 'USE ADDRESS MARK' ANSWER 'Y' AND THEN TYPE IN THE ADDRESS MARK WHICH YOU NOTED DOWN FOR THE RANGE OF TRACKS TO BE BACKED-UP. SIMPLY PRESS RETURN TO THE REST OF THE QUESTIONS AND THEN RETURN TO THE MAIN MENU. SELECT 'N' TO ENTER NIBBLES AWAY][, AND ANSWER 'Y' TO THE QUESTION 'CHANGE DEFAULT OPTIONS'. USE THE <RETURN> KEY TO MOVE TO THE 'START TRACK' PROMPT, AND THEN ENTER THE FIRST TRACK TO BE BACKED-UP. PRESS RETURN AND THEN TYPE IN THE LAST TRACK TO BE BACKED-UP WITH THE CURRENT ADDRESS MARKER SETTING. IF THE TRACKS IN THE SPECIFIED RANGE ARE NOT SPACED AT 1 TRACK INTERVALS, ENTER THE INTERVAL AT THE 'TRACK INCREMENT' PROMPT. PRESS RETURN FOR THE FOLLOWING QUESTIONS AND BEGIN THE BACKUP AFTER INSERTING THE DISKS WHEN PROMPTED. WHEN YOU RETURN TO THE MAIN MENU, REPEAT THE ABOVE PROCEDURE FOR EACH RANGE OF TRACKS WHICH CONTAINS A DIFFERENT ADDRESS MARKER.

NOW COMES THE MOMENT OF TRUTH! TRY TO BOOT UP THE BACKED-UP DISK (IF THE ORIGINAL HAD A WRITE-PROTECT TAB, THE BACK-UP SHOULD TOO!). IF THE BACKUP BOOTS, THEN ALL WENT SUCCESFULLY.

STEP 3:

IF THE BACK-UP DID NOT WORK PROPERLY THEN THERE ARE A FEW THINGS TO LOOK FOR.

1....DID ALL OF THE TRACKS WHICH SHOULD HAVE BACKED-UP DO SO? THIS CAN BE SEEN WHILE THE BACK-UP TAKES PLACE AS A 'Y' OR AN 'N' UNDER THAT TRACKS STATUS LOCATION. IF SOME DID NOT, THEN THE ADDRESS MARKER WAS PROBABLY NOT DETERMINED PROPERLY. IF THIS IS THE CASE, THEN GO BACK TO THE TBE AND TRY THOSE TRACKS AGAIN.

2....IF EVERYTHING SEEMED TO GO WELL, BUT THE BACKUP REFUSES TO WORK (YOU MAY WANT TO TRY THE PROCEDURE AGAIN, MAYBE WITH THE SOURCE AND DESTINATION DRIVES REVERSED, TO MAKE SURE IT WAS NOT A POWER GLITCH OR OTHER SUCH

OCCURANCE WHICH MESSED THINGS UP) THE NEXT STEP IS TO TRY THE PROCEDURE WITH THE 'SYNCHRONIZED COPY' OPTION SELECTED. DISKS WHICH USE THIS METHOD OFTEN MAKE VIOLENT HEAD MOVEMENTS DURING THEIR BOOT PROCEDURE. THIS CAN BE A CLUE TO THIS TYPE OF PROTECTION.

ADDITIONAL INFORMATION:

ON SOME DOS 3.3 DISKETTES, THE GAPS BETWEEN THE SECTORS ARE REDUCED IN SIZE. IN SOME CASES THEY CAN BE AS SMALL AS 4 OR 5 BYTES. WHEN NIBBLES AWAY][FINDS THE BEGINNING OF A SECTION OF DATA, IT NORMALLY ADDS 8 BYTES OF SYNC JUST BEFORE THE DATA. THIS WILL NORMALLY PUT SYNC BYTES INTO THE GAP BEFORE THE DATA, WHERE IT SHOULD BE. HOWEVER, IF A DISK HAS VERY SMALL GAPS, THEN THE ADDED SYNC CAN OVERWRITE THE END OF THE PREVIOUS SECTOR. THE PARAMETER FIX AMNT CONTROLS THE NUMBER OF SYNC BYTES WHICH ARE ADDED, SO THIS VALUE CAN BE REDUCED TO PREVENT ANY DATA FROM BEING OVERWRITTEN. THE VALUE THAT NIBBLES AWAY][USES FOR THE SYNC WHICH IT PUTS IN IS CONTAINED IN THE PARAMETER FIX VALU. NORMALLY THIS IS A $7F, BUT IT CAN BE SET TO ANY DESIRED VALUE.

IT SHOULD BE NOTED THAT NIBBLES AWAY][REGARDS ANY DATA BYTE WHICH HAS ITS HIGH BIT CLEARED TO BE A SYNC BYTE. SO THE $7F WHICH IS NORMALLY IN THIS PARAMETER MEANS THAT A SYNC $FF IS TO BE ADDED. IF THE 'OVERIDE STANDARDIZER' OPTION IS SELECTED, THEN NIBBLES AWAY][WILL NOT ADD ANY BYTES, IT WILL SIMPLY CONVERT THE DATA WHICH IS PRESENT BEFORE A SECTOR INTO SYNC, WITHOUT CHANGING ITS VALUE. THIS TECHNIQUE CAN ALSO BE USED FOR DISKS WHOSE GAPS ARE VERY SMALL.

ANOTHER ITEM TO WATCH FOR IS DISKS WHOSE TRACKS APPEAR TO BE VERY LONG. SOME DISK PROTECTION SCHEMES PUT GARBAGE ON A PORTION OF THE TRACK. WHEN THIS GARBAGE IS READ BACK, MORE BYTES ARE READ IN THAN WERE WRITTEN OUT. THIS CAUSES THE TRACK TO BE LONGER THAN NORMAL, AND IN SOME CASES IT BECOMES SO LONG THAT THE DEFAULT PARAMETERS FOR NIBBLES AWAY][CANNOT FIND THE DATA PROPERLY. THE PARAMETERS DATA MIN AND DATA MAX CONTROL THE MINUMUM AND MAXIMUM TRACK LENGTHS (IN INCREMENTS OF 256 BYTES) WHICH NIBBLES AWAY][WILL ACCOMODATE. THE NORMAL VALUE OF DATA MAX IS $1D, BUT THIS CAN BE SET TO A HIGHER VALUE, SUCH AS $25, IF A TRACK APPEARS TO BE VERY LONG. EVEN THOUGH THE TRACK MAY READ IN AS A LARGE NUMBER OF BYTES, MANY OF THESE WILL BE REMOVED BY THE NIBBLE FILTER, SINCE THEY ARE GARBAGE BYTES. THIS WILL ASSURE THAT THE AMOUNT OF DATA WRITTEN BACK OUT WILL NOT BE TO LARGE TO FIT ON THE DESTINATION TRACK.

WHEN NIBBLES AWAY][FINDS A SECTOR OF DATA, IT LOOKS FORWARD IN THE DATA TO FIND A SECOND OCCURANCE OF THE SAME PATTERN. THIS INSURES THAT THE SECTOR HAS BEEN READ IN AND LOCATED CORRECTLY. ON MANY DISKS, THERE IS A PRIMARY SECTION OF DATA, CALLED THE ADDRESS FIELD, AND THE THE ACTUAL DATA FIELD FOLLOWS. IN BETWEEN THESE IS A SMALL GAP, AND MANY TIMES IT CONTAINS RANDOM INFORMATION. THIS MEANS THAT NIBBLES AWAY][SHOULD ONLY MATCH THE NUMBER OF BYTES WHICH ARE FOUND IN THE ADDRESS FIELD, SINCE THE BYTES IN THE GAP MAY NOT READ AS THE SAME VALUE EVERY TIME. THE PARAMETER FIND MAX CONTROLS THE NUMBER OF BYTES WHICH ARE CHECKED DURING THIS PROCEDURE. THE DEFAULT VALUE OF $0C WORKS IN MOST CASES, BUT SOME DISKS USE A SMALLER ADDRESS FIELD WHICH MAY REQUIRE THIS PARAMETER TO BE SET TO A SMALLER VALUE. HOWEVER, IF THIS PARAMETER IS SET TOO LOW, THEN NIBBLES AWAY][MAY IDENTIFY THE MATCH FOR A SECTION OF DATA WHOSE FIRST FEW BYTES ARE THE SAME, BUT WHICH DIFFER LATER ON. THEREFORE ONE SHOULD EXCERSIZE CAUTION WHEN LOWERING THIS VALUE.

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK,HERE TIS SOME BASIC SATELITE TELCO
TUTORIALS NEVER BEFORE WRITTEN!

FIRST OF ALL EVERY SATELITE HAS 24
TRANSPONDERS EACH 36 MHZ WIDE.
INDIVIDUAL TELCO CARRIERS ARE 4KHZ
WIDE.THE VOICE/DATA CARRIER IS USED
TO MODULATE A DOUBLE BALANCED MODULATOR
WHERE ONE OF THE 2 SIDEBANDS TIS ELIMIN
ATED WITH A FILTER.THE REMAINING SIDE
BAND SIGNAL IS APPLIED TO ANOTHER
CARRIER FREQUENCY BETWEEN 64-108 KHZ.
 THESE CARRIERS ARE THEN MULTIPLEXED
TOGETHER IN GROUPS OF 12.SUPERGROUPS
CONTAIN 5 GROUPS AND MASTERGROUPS
CONTAIN 5 SUPERGROUPS.(300 CARRIERS)
 THESE ARE THEN SENT VIA SATELITE IN
"PACKETS" CONTAINING EITHER GROUPS,
SUPERGROUPS,OR MASTERGROUPS IN THE
0 TO 10.75 MHZ RANGE ON A TRANSPONDER.
MASTERGROUPS ARE 5 SUPERGROUPS MULTIPLE
XED AND 1 MIXING CARRIER PER SUPERGROUP
 WHICH ARE UPLINKED BY THE TOC(TOLL
OPERATIONS CENTER) LOCATED IN VARIOUS
AREAS OF THE U.S.
 BLOCK CONVERSION IS USED TO EXTRACT
GROUPS DURING DOWNLINKING.

CONTINUED NEXT MSG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THEN THE GROUP IS MICROWAVED VIA
TERRESTIAL MICROWAVE CIRCUIT TO THE
DESTINATION TOC WHICH DEMODULATES THE
GROUP USING A LOWER SIDEBAND RECEIVER.
THE CARRIERS ARE THEN SENT TO THEIR
FINAL DESTINATION VIA LEASED TELCO
LINE OR RADIO CIRCUIT.

SCPC (SINGLE CHANNEL PER CARRIER)
MAY OPERATE BY THEMSELVES OR BE SLTTED
(OOPS)SLOTTED NEXT TO GROUPS.THESE ARE
60 KHZ WIDE WITHIN 65 TO 85 MHZ

AS SMALL AS AN 4.5 METER DISH WITH 30-
100 WATTS POWER WILL ACHIEVE UPLINK
CAPABILITIES.

TVRO RECIEVES 3.7-4.2 GHZ AND DOWNCONVE
RTS TO SOME IF(SUCH AS 70MHZ) THEN
DEMODULATES TO 0-10.75 MHZ(BASEBAND)
IF YA OWN A TVRO-RUN CABLE FROM THE
VIDEO/DEMODULATED/BASEBAND OUTPUT OF
THE RECEIVER TO A HAM RECEIVER (SUCH
AS AN ICOM R-71A) TO TUNE IN THE
0-10.75 MHZ RANGE OF YOUR SATELITE

OF CHOICE.HEE-HEE-HEE

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK-NOW YA GOTTA FIND OUT WHERE TO LOOK
FOR TELCO TRANSPONDERS.BELOW TIS YE
MAIN SATELITE/TELCO INFO-

SATELITE	TELCO TRANSPONDERS
SATCOM V	3-5-7-11-17-13
SATCOM IR	5-7-9-10-17-23
COMSTAR IV	1-3-4-6-7-15-16-19-22-23
WESTAR IV	14-20-24
TELSTAR IIIA	ALL
COMSTAR III	2-5-6-7-9-14-15-16 18-20-21-22-23
WESTAR II	1-4-5-8-9
GALAXY II	12 MCI TRANSPONDERS HERE
COMSTAR 01/02	ALL
SATCOM IIR	3-4-7-19-21-22-23-

REMEMBER EACH CARRIER MAY USE TO 2700
VOICE CHANNELS WITH NUMBERS GROWING.
DUPLEX FM OR SSB/SCPC CARRIERS ARE
YE FUTURE PHREAKERS TARGETS.

 TRYING TO TRACE YE NEW GENERATION OF
SATELITE PHREAKS WILL LEAD TELCO SECUR
ITY STRAIGHT TO A LOCATION IN OUTER
SPACE!!! HAR-HAR-HAR

NUFF SAID-
BOOTLEG

DUE TO POPULAR DEMAND AND THE HUGE
AMOUNT OF INFORMATION NOW AVAILABLE
TO ME,I HAVE DECIDED TO PUBLISH A
SISTER MAGAZINE TO THE BOOTLEGGER
CALLED-

"THE HACKER"

SAME SUBSCRIPTION PRICE AS THE
BOOTLEGGER.SAME ADDRESS ALSO,BUT
THE HACKER WILL BE PUBLISHED IN
BETWEEN BOOTLEGGER ISSUES SO THAT
YOU CAN GET INFO A LOT QUICKER!
 NATURALLY THE HACKER WILL PUBLISH
A LOT OF GREAT INFO PERTAINING TO
THE UNDERGROUND HACKING WORLD-
SUBSCRIBE NOW-DON'T MISS ISSUE #1.

(SOME OF THE HACKERS INFO WILL

INCLUDE FILES TAKEN RIGHT OUT OF
THE LATEST ESS MANUALS!)

NUFF SAID-
BOOTLEG

```
-------------------------------------
         FUN STUFF FOR SYSOPS
---------------------------------------
```

First, you must be a sysop.
(Obviously!) Or, you may be at a
sysop's house (When he or she is not
around.)

Second, you must be VERY popular,
or VERY daring. Either way, your
victim will have a strong dendency
to: a) crash your board, b) hate you,
or c) spread malicious rumors about
you, and, or your board to everyone
in the world that will listen.

I am going to write about AE
fun first, and then Net-Worx.

```
                AE Fun
                -- ---
```

So you are bored, and want to
have some fun, huh?
Go into your room, or wherever
you have your apple, and sit down.
Turn on the monitor, and lets see
if there is a leech on the line.
(-note: if you are the unlucky type,
I suggest that you give this up,
because for all you know, that sysop
of the 20meg board is on your line,
and he's going to be your victim!!)
Now for some of these pranks, you
will need to make things before-
hand. I suggest you read this through,
and make the necessary mods.

1) This one is probably my meanest
trick, and should only be used on
people like Matt Ackerett, or Little
Al.
Your victim has to be leeching
a game off of your AE for this to
work.
You wait until your victim is at
his last 2 blocks of memeory to go
until the transfer is done, and
you take out the disk.
This will ruin the >entire<
transmission. It won't piss them off
too bad if it is only 50 or so blocks,

but can you imagine:

```
Send: Matt Ackerett is a fag
290 blocks
crc=167
<289>
```

    Note- The victim has to get 290
blocks, you only let the victim get
289!

   At that point, take out the disk!
They have just waited 1/2 hour for
nothing! They can't get the last block
and have to go through the whole
thing again!! Ha ha!
   This is very mean, especially if
they aren't phreaking, they have
been >paying< for it all!

2) If you want to see if the person
on is intelligent...simply let him
catalog your drive once, then when he
is done, take the disk out, and
put in the disk from the other drive.
When they catalog the disk next, it
will be different!
   This will freak them out, they will
think that they have switched to d1
somehow. The victim will then L)og the
drive, and find it still on D2. Wow!
   Hopefully they will catalog D1
anyway, thinking that they were
originally on D1 and it switched.
Now comes the fun.
   Put the right disk back in D2,
and put the disk that used to be in D2
into D1, so they will get the same
catalog.
   Now they are confused. Now
they will catalog D2, and find the
normal stuff. Hopefull they will
read something, now take the disk
out while they are typing in the name,
and slip the other disk in. It
will say 'file not found.'
   Good. Now they will catalog it,
and look! The wares have changed!
Now something is wrong here! They
will say:

hey! stop it!

   Oh no! They are on to your scheme!
But, 1 last joke! Get a copy-protected
type disk, one that you <gasp> bought.

They won't be able to catalog this
at all! Ha!
   If they get mad, they might
say something like:

        Hey! Stop it!

   But will you listen? nnnoooooo!
Take the disk out, and slip something
totally new, preferably the disk that
has "sneakers" or some ancient wares.
Maybe they will think these are the
latest! Watch them post!:

      Hey! I just got some new
      Warez! Do you want to trade??

hah hah!

  Satisfied, you may put the normal
disks back in and walk off to see
some football game.

3) Lock out the space-bar. This will
make it so that they can't type a
<space>. Then, they can't read
anything that requires a space.
Most likely the victim will think that
there is something wrong with >his<
computer. Thusly sending him/her/it
into a 1/2 hour scan of their install
program to see what is wrong.

4) Change the commands...such as:

```
d)irectory= c)irectory
-           -
```

   They will have to hack at the
commands! This won't be too funny,
because they won't do anything stupid
like posting:

     hey your commands are screwed!

   Most likely they wont find the
command for 'copy'.

5) lock out the "ctrl-c". This will
piss them off when the victim just
can't exit from posting. Ha!

6) Change the ring count, most, or
almost >all< AE lines are set to
pick up after just 1 ring. Change it
to...say...5 rings, and only tell your
friends that it is at 5 rings. When

they call, they will only wait for
about 2 rings, and hang up thinking
that the line is down. Only the people
you like will get on, because they
will be the only ones to wait 5 rings.
Mean huh?

7) When someone is posting, or c)opy-
ing a message, pick up the voice line,
and blow into the reciever. This will
put all of these weird characters onto
the screen. He will save a gay looking
message, that will make it look like
the victim can't type!!

Net Works

--- -----

I don't have as many fun tricks
with net-worx as I do with AE, but
here are a couple of my favorites...

1) In the program, make a bug, like
"ctrl-k" that when pushed (like ctrl-t
for chat) it will dump you into basic.
take out the disks, and put in like
the "bare-bones" net-worx disk and let
them have fun reading fake messages,
mail, and passwords. Ooooh! They will
think:

oh yay! I have everyone's pass!

Now, see if he/she will init the
disks, if they do, you know what type
of user it is. If they are nice,
and 'hang' the line for you so that
no one wil be able to get on after,
or they try to beep you, then give
them a level raise.

2) Be a tyrant. Juggle their levels
while they are on. Like break into
chat, change their level, and watch
them get all mad.

3) Break into chat, and just walk off,
leaving a frustrated user sitting
there.

4) break into chat, and change the
time. In other words, leave them
with -10 minutes, instead of 35 or
so.

5) when they log off, and they get
that stupid message about:

```
            Thank you for
              calling

and all of that, press 'ctrl-c' a
few times, and they will be brought
back. Wow! What happened? Let them
try to log off a few times nd keep
pressing ctrl-c. Finally they should
just press 'reset'. He he!


    I hope you have enjoyed these
little pranks. Your users will hate
you if you do this too often, unless
they are like Matt Ackeret or Little
Al. Then it doesn't matter much.
    Remember! I hold no responsibilty
for people wanting to crash your
system because they are so pissed
at you!

Sysop fun- A Surf Rat file.

Call The Realm of the Rogues!
             415/941-1990  20 megs!!

Call The Twilight Zone!
             408/253-2140  C00L!

Call The Gossip Line! (AE)
             415/949-1049:pw/gossip


And hey! dont put >your< name in here!

Surf..
 -BFB

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

A LOT OF YOU HAVE BEEN ASKING FOR
PROGRAMS THAT WILL HACK OUT VARIOUS
CODES,NUMBERS,PSWDS,ETC.

OK-IVE COMPILED THE MOST POPULAR
AND EFFICIENT HACKING PROGRAMS
EVER ASSEMBLED!THESE INCLUDE SUCH
INFAMOUS PROGRAMS AS THE OUTLAWED
"TSPS" AND THE NOTORIOUS "JOSHUA".
ALONG WITH THESE FAVORATES,INCLUDED
ARE THE 600 CODE PER NIGHT HACKING
PROGRAM BY THE PROFESSOR.ALSO,ALL THE
OTHER UNDERGROUND HACKING PROGRAMS
THAT HAVE EARNED THEIR FAME IN THE
SPIRIT OF WARGAMES!!!

TO ORDER "THE HACKER" SEND $100
```

```
TO-

          THE HACKER

        1080 HAYS CUT-OFF ROAD
        CAVE JCT.OR.97523

NUFF SAID-
BOOTLEG

P.S. THIS COLLECTION OF HACKING
     PROGRAMS WILL DEFINATELY TAKE
     UP SEVERAL DISKS OF SPACE!
THE BOOTLEGGER HAS A FOOLPROOF METHOD
OF SAFELY TRADING DISKS WITHOUT BEING
RIPPED OFF!
SIMPLY SEND 10 OR MORE DISKS TO ME
WITH $2 TO COVER POSTAGE,AND I WILL
HOLD THEM UNTILL THE PERSON YOU ARE TRADING WITH ALSO SENDS THE DISKS YOU
WANTED! WHEN BOTH PARCELS ARE RECEIVED-I'LL
MAIL THEM OUT.IF ONLY ONE PARCEL IS RECEIVED- AFTER 2 WEEKS ILL MAIL IT BACK,OR
 FILL YOUR DISKS WITH NEW PROGRAMS!
I RESERVE THE RIGHT TO COPY ANY PROGRAMS WHILE WAITING! HEE-HEE

NUFF SAID-
BOOTLEG

P.S. AT LEAST ONE PARTY TO THE TRADE
     MUST BE A CURRENT SUBSCRIBER!
ALSO-FILL BOTH SIDES OF YOUR DISKS.
I'VE BEEN GETTING SOME OLD STUFF
IN THE TRADE CLUB LATELY,SO WHAT IM
DOING IS EXCHANGING OLD FOR OLD,NEW
FOR NEW! (GET THE HINT?)

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

VA (VARIABLE ANI ROUTE TREATMENT) IS
USED TO PROVIDE THE START SIGNALS AND
CATAGORY SIGNALS AS REQUIRED FOR
VARIOUS PULSING FORMATS,SUCH AS BELL
SYSTEM STANDARD AND NT-500.THE SYSTEM
OUTPUT AND INPUT PARMS FOR THIS ROUTE
TREATMENT ARE-

ANIFST & ONIST

 START SIGNALS FOR AN ANI/ONI FAIL TYPE
CALL ARE 15 FOR KP,12 FOR ST,13 FOR
STP,14 FOR ST2P,11 FOR ST3P,OR 0 FOR
SENDING THE START SIGNAL PASSED BY THE
TRANSLATOR.

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
```

DATE POSTED:

WANT DTMF DECODER FER YER COMPUTER?

THEY CAN BE HAD FROM $22.95 TO $89.95
FROM ENGINEERING CONSULTING AT
714-671-2009

LOTS OF PHUN WITH YE STUFF THIS COMPANY
SELLS.ASK FOR CATALOG

OH YEA- VISA AND MASTERCARD ACCEPTED!

             HAR-HAR-HAR

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK,HERE TIS SOME BASIC SATELITE TELCO
TUTORIALS NEVER BEFORE WRITTEN!

FIRST OF ALL EVERY SATELITE HAS 24
TRANSPONDERS EACH 36 MHZ WIDE.
INDIVIDUAL TELCO CARRIERS ARE 4KHZ
WIDE.THE VOICE/DATA CARRIER IS USED
TO MODULATE A DOUBLE BALANCED MODULATOR
WHERE ONE OF THE 2 SIDEBANDS TIS ELIMIN
ATED WITH A FILTER.THE REMAINING SIDE
BAND SIGNAL IS APPLIED TO ANOTHER
CARRIER FREQUENCY BETWEEN 64-108 KHZ.
 THESE CARRIERS ARE THEN MULTIPLEXED
TOGETHER IN GROUPS OF 12.SUPERGROUPS
CONTAIN 5 GROUPS AND MASTERGROUPS
CONTAIN 5 SUPERGROUPS.(300 CARRIERS)
 THESE ARE THEN SENT VIA SATELITE IN
"PACKETS" CONTAINING EITHER GROUPS,
SUPERGROUPS,OR MASTERGROUPS IN THE
0 TO 10.75 MHZ RANGE ON A TRANSPONDER.
MASTERGROUPS ARE 5 SUPERGROUPS MULTIPLE
XED AND 1 MIXING CARRIER PER SUPERGROUP
 WHICH ARE UPLINKED BY THE TOC(TOLL
OPERATIONS CENTER) LOCATED IN VARIOUS
AREAS OF THE U.S.
 BLOCK CONVERSION IS USED TO EXTRACT
GROUPS DURING DOWNLINKING.

CONTINUED NEXT MSG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

THEN THE GROUP IS MICROWAVED VIA
TERRESTIAL MICROWAVE CIRCUIT TO THE

DESTINATION TOC WHICH DEMODULATES THE
GROUP USING A LOWER SIDEBAND RECEIVER.
THE CARRIERS ARE THEN SENT TO THEIR
FINAL DESTINATION VIA LEASED TELCO
LINE OR RADIO CIRCUIT.

SCPC (SINGLE CHANNEL PER CARRIER)
MAY OPERATE BY THEMSELVES OR BE SLTTED
(OOPS)SLOTTED NEXT TO GROUPS.THESE ARE
60 KHZ WIDE WITHIN 65 TO 85 MHZ

AS SMALL AS AN 4.5 METER DISH WITH 30-
100 WATTS POWER WILL ACHIEVE UPLINK
CAPABILITIES.

TVRO RECIEVES 3.7-4.2 GHZ AND DOWNCONVE
RTS TO SOME IF(SUCH AS 70MHZ) THEN
DEMODULATES TO 0-10.75 MHZ(BASEBAND)
IF YA OWN A TVRO-RUN CABLE FROM THE
VIDEO/DEMODULATED/BASEBAND OUTPUT OF
THE RECEIVER TO A HAM RECEIVER (SUCH
AS AN ICOM R-71A) TO TUNE IN THE
0-10.75 MHZ RANGE OF YOUR SATELITE
OF CHOICE.HEE-HEE-HEE

NUFF SAID-
BOOTLEG

MSG LEFT BY: SYSTEM OPERATOR
DATE POSTED:

OK-NOW YA GOTTA FIND OUT WHERE TO LOOK
FOR TELCO TRANSPONDERS.BELOW TIS YE
MAIN SATELITE/TELCO INFO-

SATELITE	TELCO TRANSPONDERS
SATCOM V	3-5-7-11-17-13
SATCOM IR	5-7-9-10-17-23
COMSTAR IV	1-3-4-6-7-15-16-19-22-23
WESTAR IV	14-20-24
TELSTAR IIIA	ALL
COMSTAR III	2-5-6-7-9-14-15-16 18-20-21-22-23
WESTAR II	1-4-5-8-9
GALAXY II	12 MCI TRANSPONDERS HERE
COMSTAR 01/02	ALL
SATCOM IIR	3-4-7-19-21-22-23-

REMEMBER EACH CARRIER MAY USE TO 2700
VOICE CHANNELS WITH NUMBERS GROWING.
DUPLEX FM OR SSB/SCPC CARRIERS ARE
YE FUTURE PHREAKERS TARGETS.

TRYING TO TRACE YE NEW GENERATION OF
SATELITE PHREAKS WILL LEAD TELCO SECUR
ITY STRAIGHT TO A LOCATION IN OUTER
SPACE!!! HAR-HAR-HAR

NUFF SAID-
BOOTLEG

DUE TO POPULAR DEMAND AND THE HUGE
AMOUNT OF INFORMATION NOW AVAILABLE
TO ME,I HAVE DECIDED TO PUBLISH A
SISTER MAGAZINE TO THE BOOTLEGGER
CALLED-

"THE HACKER"

SAME SUBSCRIPTION PRICE AS THE
BOOTLEGGER.SAME ADDRESS ALSO,BUT
THE HACKER WILL BE PUBLISHED IN
BETWEEN BOOTLEGGER ISSUES SO THAT
YOU CAN GET INFO A LOT QUICKER!
NATURALLY THE HACKER WILL PUBLISH
A LOT OF GREAT INFO PERTAINING TO
THE UNDERGROUND HACKING WORLD-
SUBSCRIBE NOW-DON'T MISS ISSUE #1.

(SOME OF THE HACKERS INFO WILL
INCLUDE FILES TAKEN RIGHT OUT OF
THE LATEST ESS MANUALS!)

NUFF SAID-
BOOTLEG

```
=============================================================================
DOCUMENT catfur.app
=============================================================================

                         **** Cat-Fur ****
                   Disected By -:-:Freq Freak:-:-
             With help from:  The Highflier / Bit Blaster
                 Rock'n Roll Harbour 10 meg BBS/Catfur
                    [305] 557-8778  300/1200 baud
                ++++++++++++++++++++++++++++++++++++++
Notes:

For Online -> Poke 2046,acc lvl
            Poke 2047,BB
            Brun Cat-Fur

  @ACC -> Text File on drive Contains access lvl required to access it.
  @FUR -> Applesoft file to be run after hung up in online
  CAT.HELLO -> Welcome file to be read on remote logon
                ++++++++++++++++++++++++++++++++++++++

1000 - Move 3rd Text line to 280-2A8
100D - Set Program Pointers
103E - Move 1000-4A00 to 6000-9A00
1064 - Goto prog at $6209

6067-6208 - Modem S/R's JMP Table
           6123-Send Byte
           6126-Ck Carrier
           6129-Read data
           612C-Com Byte
           612F-Pick up Phone
           6132-Set 103/orig
           6135-Set 212 answer
           6138-Hang up
           613B-Dial # in acc
           613E-Setup Modem regs
           6141-Set 103/ans
           6144-Ring Detect

6209 - Init vars outside of prog
6287 - Check for online run
628F - Set misc vars
62A5 - Cls, and output main menu
65AC - Checksum?
65D1 - Fix screen and setup modem
65FD - Set carrier type
6600 - Update Stats, Ck ring, Ck key
6613 - Get Key and Jump accordingly
665A - Ctrl/C - Exit
6677 - P      - Phone toggle
6685 - M      - Modem Mode toggle
6693 - D      - Dos Command
66EC - Output X of char at $66ED
66F5 - C      - Change Drives
6845 - Output Vol in 3 digit #
685D - Ctrl/T - Toggle force D1 trans.
```

```
6874 - Ctrl/N - Toggle Hard Drive
6891 - Fix inputted line
68AF - R      - Reconfigure
6A73 - Update CH-CV S/R
6A7E - Update status windows S/R
6B20 - Wait S/R
6B36 - Setup Modem&Carrier type S/R
6B46 - Check For <Esc> S/R
6B54 - Await Carrier or Esc S/R
6B6E - E      - Get Carr, goto Cat-Fur
6BD6 - Beep S/R
6BE4 - Set some Dos vectors
6C0E - 'Error' S/R
6C54 - Output string S/R
6C8E - Output a Char to screen
6C98 - Upper case conversion
6C9F - Print X,A then Cr
6CA8 - Totally Useless to know...
6CB1 - Set flags, 280-2A8 to 500-528
6CDC - Get key if there, convert it
6D1F - Ck byte read from key & output
6D4A - Ck byte read from Modem       "
6D75 - Print $32C,$313
6D95 - Print $31B,$31C
6DB5 - Err Message
6DFD - Disable Interrupt
6E08 - Awaiting Handshake Msg
6E2D - Handshake Received Msg
6E52 - Receive Handshake
6EB2 - Send Handshake
6F07 - Set 202 Receive
6F25 - Set 202 Transmit
6F3E - Set Interrupt
6F51 - Set Carriers, XR on 300 Baud
6F6C - Interrupt Routine
6F89 - Xmit Aborted Routine
6FD9 - Screen for Transfer Status
7150 - Output massive amounts of -'s
717D - Clear mem S/R
7197 - Clear mem S/R
71B5-7278 - Send full disk
727B-7374 - Receive full disk
7397-7419 - Read Sector etc...
741C-7442 - Error in rwts msg
7445-74F8 - Receive Data
74F9-7715 - Real big mess. Transmit?
7718-772A - Ck key & stuff
772B-77F6 - Send data
77F9-7830 - S/R
7831-78E2 - S/R
78E3-795B - S/R
795C-796B - S/R
796C-798E - Move cursor, Cout S/R
798F-79B1 - Move cursor, Cout S/R
79B2-79F9 - Ck key, Sta, Cout etc. S/R
79FC-7A1D - Inc buff, Cout S/R
7A1E-7AD8 - Read file
7ADB-7B85 - Write file S/R
```

```
7BB6-7B9B - S/R (End of trans misc.)
7B9E-7C38 - Transfer complete routine
7C39-7C5D - Sound output S/R
7C5E-7D3A - Open file etc...
7D3D-7DCE - Setup lookup table ?
7DD1-7DDC - Call DOS File manager
7DDD-7DEB - Set 31B,31C,31A,31F to #00
7DEC-7E7F - Transfer buff ck&move ?
7E80 - Slot & vol store misc.
7E98 - Inc Byte at 77,78 if page, pop
7EA1 - Add number to byte at 77,78
7EAF - Misc. manipulation
7EEE - S/R
7F43 - S/R
7F86 - Swap buff locs ($0500/$0200)
7F95 - Do $79B2 5 times
7FBB - (A EOR $0319) + A
7FC8 - Select files routine
804B - S/R
8058 - Output spaces S/R
807C - Transfer Menu S/R
8175 -
8291 -
82C9 -
839B - Store CH-CV
83A6 - Restore CH-CV
83B2 - Cat-Fur Transfer Section
83BE - Cls & print display
84C6 - Ck Carrier, Enter menu
84D8 - Get & process modem byte
84FD - Get & process Key pressed
851D - Esc Pressed.
85A3 - Ck key hit and do Jsr's
860E - Ck byte sent & do Jsr's
86EA - Lost Carrier, Do second ck
8705 - Lop-sided send-Local
8708 - Lop-sided get -Remote
872F - Lop-sided get -Local
8732 - Lop-sided send-Remote
8751 - Both Transfer -Local
8754 - Both Transfer -Remote
877F - Send Catalog  -Local
8782 - Receive Cat   -Remote
8909 - Receive Cat   -Local
890C - Send Catalog  -Remote
8A0A - Set Drive
8A1D - Clear some mem
8A3E - Hang up
8A4A - S/R
8AC7 - S/R
8B09 - S/R
8B4B - S/R
8BC0 - S/R
8BDB - S/R
8C1A - S/R
8C3F - S/R
8C6D - S/R
8C89 - S/R
```

```
8CAB - S/R
8CC7 - Terminal Mode
8D69 - Get key    - terminal
8D76 - Jump to terminal command s/r
8DA0 - Terminal '?' command
8E5A - Terminal 'K' Toggle chat
8E87 - Terminal 'I' Dos Command
8E9D - Terminal 'H' Hang up
8ECF - Terminal 'D' Dial
8EF9 - Get # to dial
8F50 - Dial # in buffer
8F8C - Await Carrier
8FEC - Redial if '/' found
9000 - Carrier detected
905D - Terminal 'E' Enter Catfur
9082 - Terminal '-' command
9165 - Lost carrier
91AF - Terminal '+' unattended
9219 - Run @FUR if lost carrier
9268 - Wait call
92C3 - Wait carrier
9309 - Carrier Detected
9325 - Get password if exists
9362 - Hang up if wrong
9380 - Jmp $9754 sometimes
9383 - Remote Prompt '(>'
93A0 - Get key
93A7 - If Ctrl/K enter chat
93EA - Check key hit
93FF - Do jsr's for key
9410 - Remote '?' command
941A - Remote 'H' hang up command
9450 - Remote 'D' directory command
948B - Input from Screen and Modem
94BF - Remote 'L' log drive command
9587 - Abort access check
958C - Ck access to drive
95E8 - Access Denied
9609 - Access Permitted
9611 - Search F-name for char in A
9625 - Move F-name to key buff
9640 - Change slot # to A s/r
9647 - Update Volume # s/r
9654 - Change Drive # to A s/r
965B - Check Slot if valid
9674 - Clear text buff to A0's
9680 - Remote 'V' view text command
9719 - Print filename S/R
973C - Reset I/O Ptrs & "Ctrl/d Close"
9754 - Move Welcome f-name & call View
9778 - Free space on disk S/R
97B5 - Remote 'E' enter transfer
97C9 - Terminal 'L' log drive
97D3 - Terminal 'X' Exit Terminal mode
97DB - Dial Autosearch
97F7 - Reset output pointers to $6C8E (screen only)
980C - Reset output pointers to $9896 (screen & modem)
9817 - Reset input pointers to $FF58 (Rts)
```

```
9822 - Reset input pointers to $948B (Screen & modem)
982D - Send esc,1,jsr $6F51,2,2,2
9855 - Send esc,2,jsr $6F25,2,2,2
987D - Output A with cursor
9896 - Output to modem and screen
98A5 - Out $(X,A) til #$00
98B9 - Jmp Data for terminal
98D8 - Jmp Data for Remote
98EB - Text 'Welcome to Cat-fur etc...'
9929 - Text 'Password:'
9934 - Text for Remote menu '?' cmd
9981 - Text 'Directory...'
998F - Text 'Entering Transfer Section'
99AC - Text 'Access Denied!'
99BD - Text 'View:'
99C4 - Text 'Current:'
99CE - Text '    New:'
99DA - Text 'Hang Up (Y/N):'
99EA - Text 'CAT.HELLO'
99F5 - Shift mod 00 if none, else 01
99F6 - Text containing password
99FC - Text 'AT'
99FF - If #$FF then unattended
        #$00 hangs up after transfer

                  :| Brought to you by Bit Blaster |:
```

==
DOCUMENT catstuff.app
==

Uploaded By: RAMPANT CRIMINAL
%=%

Expanding your Apple Cat //
By:
((%>> The Ware-Wolf <<%))
(Hi-Res<>Hijackers/The 202 Alliance/WareBusters!)

%=%

THE PIPELINE..BBS/CATFUR 300/1200
OVER 10MEGZ <718> 351 5678

The Apple Cat // modem is by far the most expandable modem on the market today. Of course it's also the choice modem of pirates because of it's inexpensive half-duplex 1200 baud capabilities. The expansion module available for the cat has several very useful functions. Rather than shelling out $30 bucks for one which you may only use a few of the features this file tells you how to build just certain features or even the whole package.

First off you'll need some basic knowledge and tools. As for the knowledge you'll need to know how to solder pretty well, you'll also proabibly have to know DC from Hz and +12V from RS232. Ok now, If you can handle that that, you'll need these tools:

- A soldering iron and solder
- A flat, 14 wire, female cable. Preferably multi-colored.
    * Note: Single strands of wire will do but they risk damaging your cat.

We'll be connecting the wires to the J2 connector (see owner's manual, fig. 2). Remember that there are 25 pins on this connector. Each pin numbered starting with pin 1 in the rear of your computer and pin 25 closest to the keyboard. We'll only be working with the first 14 pins. The rest are for the 212 and speech synthesizer cards.

Here is a table which tells something about each pin:

Pin #	Function	Direction	Feature
01	Transmit Data	Output	EIA-RS232C Printer interface
02	Receive Data	Input	
03	Clear to Send Signal	Input	
04	Signal Ground	GND	
05	AC line reference (60Hz)	Input	BSR Remote control
06	Signal Ground	GND	
08	+12V DC	Output	
09	120 KHz Control Signal	Output	
07	+12V DC	Output	Off-Hook LED
12	LED Drive	Output	
10	Tape Recorder Control	Input	Tape Recorder
11	Tape Recorder Control	Output	
12	Audio Signal to Tape	Output	
14	Signal Ground	GND	

----
* Note: This table corrects several errors which occur in the table in the Owner's Manual.

Bulidin' the On/Off hook indicator
==================================
Required parts: 12V DC LED
==================================
   This is the most inexpensive and simple of the projects. All you must do is connect the wire leading from pin 7 to the positive pole of the LED and connect pin 12 to the remaining pole. Solder connections firmly and whenever the modem is off-hook the LED will light.

Hooking up a tape player
========================
Required parts: Tape Recorder with adjustable record level, 3.5 mm patch cable; male on one end; stripped on the other, Patch cable with 2.5 mm plug on one end;stripped on the other.
========================
   This is proabibly the most useful feature. With this feature you may listen in on your cat. Such as when calling a board you'll never have to pick up the phone.   You also might want to do an answering machine. I'll tell you more about that later.

   To build this you must take the wires leading from pins 10 & 11 and connect them to the stripped ends of your 2.5 mm patch cable. Now take the wires leading from pins 13 & 14 and connect them to the stripped ends of your 3.5 mm patch cable. ** Note: You may have to reverse which pin goes to which wire on each cable if it doesn't work at first. Now, simply plug the 3.5 mm plug into the Mic jack on the tape recorder and plug the 2.5 mm plug into the Rem jack on the tape recorder.

   To use this you just press the Rec button(s) on your tape recorder. On most tape recorder you'll be able to hear what is going on when the modem picks up the phone. You'll notice that the tape does not move when you press record, you must do a POKE 49313,31 (Default = 0) to turn on the tape. That is how you make your answering machine. ** Note: I have included an answering machine program at the end of his file.

Bulidin` the EIA-RS232C printer interface
===
Required parts: Serial printer, RS232C cable
===
   This is pretty difficult to explain. We'll start by looking at the RS232C port on the back of your printer. This port has two rows of holes. One row has 12 holes and the other has 13.   We'll number these holes by going left to right
the first holes are 1 to 13 on the largest row, next go to the left of the smaller row and number from 14 to 25. Not all of these holes will be used.
This chart tells which wire goes to which hole:

Pin #	Hole(s)
01	12
02	11
03	19+3 (19 first)
04	07

Hooking up the BSR Remote Transformer
=====================================
Required Parts: BSR Remote Transformer
=====================================
      ** Note: This is really quite dangerous and I recommend if you wish to use this function and are unsure of your abilities that you buy an expansion module.

   Now, look at the square end of your transformer. Each hole should have a number next to it. If you don't see these numbers than just number counter-clockwise starting at the bottom left corner (notch facing the floor). There is really no good way to get the wires to stay in these holes. You may want to go to Radio Shack and look for something. Anyways be sure the transformer is not plugged into the wall and connect each pin to each hole as shown:

```
Pin #5--> Hole #3
Pin #6--> Hole #1
Pin #8--> Hole #2
Pin #9--> Hole #4
```

**Caution: Be sure that no wire touches another wire!

   To use this you must have at least one of those modules which come with the real BSR Command things. There is a program on your Com-Ware disk to control this.

**Caution: When working on these features be sure to connect them to the pins last or else damage to you or your cat may occur.

Here is the answering machine program I mentioned earlier:

```
10  REM  -> A WARE-WOLF PRODUCTION
20  POKE 49314,0: POKE 49313,0
40 S = 38142:P = 38141:M = 33056:T = 33055:C = 22357:A = 38131:D$ =  CHR$ (13)
+  CHR$ (4)
70 KB =  - 16384:PR =  - 16211:CC = 49168
80  HOME : PRINT CA
90  IF      PEEK (KB) = 195 THEN ZZ =  PEEK (CC): RUN
110  IF  PEEK (KB) = 212 THEN ZZ =  PEEK (CC): GOTO 160
120  IF  PEEK (KB) = 209 THEN  PRINT  CHR$ (8): POKE 49168,0: END
130  IF  PEEK (PR) / 2 =  INT ( PEEK (PR) / 2) THEN 90
140  PRINT "Sam:";: INVERSE : PRINT "Receiving Call": NORMAL
160  POKE 49314,2: FOR X = 1 TO 3500: NEXT
170 SA$ = "HELLO.THERE.YOU HAVE.REACHED.THE.WARE.WOLFS.COMPUTER": GOSUB 400: CA
LL A:SA$ = "HE.IS.NOT HERE.NOW.BUUT.LUCKILY.ME.AND.MY FRIENDS.ARE HERE.TO.TAKE
YOUR.MESSAGE": CALL A
180 SA$ = "NOW.LISTEN UP.SUNNY.IF.YOU DON'T.LISTEN.WE.MIGHT.HAVE TO.KICK YOUR A
SS": GOSUB 360: CALL A:SA$ = "AFTER.WE.STOP.TALKING.YOU.WILL HEAR.A.BEEP.": GOS
UB 340: CALL A
190 SA$ = "I.WON'T.HANG.UP.TILL.YOU.ARE FINISHED.LEAVING.YOUR.MESSAGE": GOSUB 3
20: CALL A
200 SA$ = "REMEMBER.TO.WAIT.FOR.THE.BEEP.": GOSUB 380: CALL A
210 SA$ = "BYE": GOSUB 300: CALL A: GOSUB 320: CALL A: GOSUB 340: CALL A: GOSUB
```

```
 360: CALL A: GOSUB 380: CALL A: GOSUB 400: CALL A:SA$ = "P...": FOR X = 1 TO 9
00: NEXT : POKE 49313,31: CALL A
220  FOR Z = 1 TO 190:V = ( PEEK ( - 16224) - 15): IF ((V / 16) / 2) <  >  INT
((V / 16) / 2) THEN  NEXT
230  PRINT Z: IF Z =  > 190 THEN 250
240  GOTO 220
250 SA$ = "THANKS FOR THE MESSAGE": CALL A
260  POKE 49314,0: POKE 49313,0
270 CA = CA + 1
280  GOTO 40
300  REM     ***ELF***
310  POKE T,110: POKE M,160: CALL C: POKE S,72: POKE P,64: RETURN
320  REM    ***ROBOT***
330  POKE T,190: POKE M,190: CALL C: POKE S,92: POKE P,60: RETURN
340  REM      ***STUFFY GUY***
350  POKE T,110: POKE M,105: CALL C: POKE S,82: POKE P,72: RETURN360  REM
   ***OLD LADY***
370  POKE T,145: POKE M,145: CALL C: POKE S,82: POKE P,32: RETURN
380  REM     ***E.T.***
390  POKE T,150: POKE M,200: CALL C: POKE S,100: POKE P,64: RETURN
400  REM    ***REGULAR***
410  POKE T,128: POKE M,128: CALL C: POKE S,74: POKE P,64: RETURN
```

To use this program first, EXEC it into basic and save it. Next boot up Sam Knobs and select the text input version. Now when run this program will put a 0 in the upper-left corner of the screen. This is how many calls you have had so far. To test the program just hit "T" to clear the call count hit "C" to quit hit "Q". It after the little greeting message it waits until there is no sound for about 6-7 seconds. So people can leave messages of unlimited length. I included the pokes for different voices so you can be creative with your messages.

```
==========
The End...
==========

              ---------------------------------------
               PIPELINE BBS/CATFUR 300/1200 10MEGS
                     <718> 351 5678
```

```
==============================================================================
DOCUMENT cheat.app
==============================================================================

   >-O N E  S T E P  B E Y O N D-<
---===-------------------------===-----

JOUSTER
BLOAD JOUSTER
CALL -151
219E:09
CTRL-C RETURN
BSAVE JOUSTER.9,A$800,L$75FF

JUMP JET     JOUSTER
131D:EA EA EA      955:EA EA EA

HARD HAT MACK      HARD HAT MACK
CTRL-L (1-3)       503:18 60 N
             50A5:EA EA EA N
             82DG

THUNDERBOMBS       CRIME WAVE
2E39:EA EA EA

NONADS                BERSERKER
41E9:EA EA EA         6179:EA EA EA
26FAG           1F00G

MONEY MUNCHERS        MILIPEED
1020:# OF MEN         602A:#
FE7G            1F00G

REPTON                REARGUARD
19C4:4C CB 19 N     CTRL-T + LEVEL #
19D7:60 N D92:EA EA

NIGHTMARE GALLERY   APPLE KONG
6818:EA EA EA         43F5:EA EA
8718:EA EA EA         C050 C057 C054
671B:EA EA EA         4000G
861B:EA EA EA

MOUSKATTACK     STAR THIEF
6A53:EA EA EA         1827:# OF SHIPS

EVOLUTION       OUTPOST
6731:# OF GUYS        2C22 & 8046:#
6000G           3798:EA EA

BELLHOP         HORIZON V
6A92:# OF MEN         5B0A:E6 (UNLIM)

STAR MAZE       BLOAD WARGLE.OBJ
50B2:EA EA      7250:# OF GUYS

SUPER PUCKMAN         CONGO
```

```
147B:04  1C40:60    5227:EA EA EA
800G            BF4G

CYCLOD                RASTER BLASTER
8025:EA EA EA         692E:EA EA EA EA EA
900G            2700G

DIG 'EM             SEA DRAGON
6EAB:FF         8C32:00 (AIR)
6D87:EA EA EA         8C59:00
5BD1:EA EA EA         8C72:EA EA (DAMAGE)
5808G           5C43G

TELEPORT (RESET)    A.E.
41D1:EA EA EA         EE1:# OF SHIPS
5F8CG           7FDG

MINER 2049'ER       THE ALIEN GAME
812:# OF LEVEL        8550:#
814:LEVEL - 1         C050 C057 C054
816:# OF MEN          800G
981G

JAWBREAKER II         BOLO
84B:# OF GUYS         14A8:EA EA
800G            1D3D:EA EA

MICROWAVE -:- PRESS RESET ON HIRES PAGE
LESS MONSTERS - 8146:00  8100G
UNLIMITED MEN - EDIT T0 SD B3E
            INSERT EA EA EA
            EDIT T19 SA B3E
            INSERT EA EA EA
UNLIMITED POWER-EDIT T19 SC B75
            INSERT EA EA EA

KAMEARI         SPY'S DEMISE
BLOAD PACK.DATA1    60AB:# OF SPYS
BLOAD PACK.DATA2    3FF1G OR
B82:EA EA   7FDG    C050 C057 C054
95C:EA EA   7FDG    1100G

TUBEWAY         DIG DUG
TRY ESC-R       5BAF:EA EA EA EA EA

SNACK ATTACK          THIEF
5B28:#        6FDG          4873:EA EA EA EA EA

PHASER FIRE     ANOTHER FOR TUBEWAY
452E:# OF SHIPS     22D5:# OF SHIPS
3FFDG           900 OR 7FD OR
                2083:EA EA

STARMAZE (ANOTHER)  SERPENTINE
459C:# OF SHIPS     81A:# OF MEN  7FDG

FREE FALL       CANYON CLIMBER
BLOAD AT A$800        2600:# OF MEN
```

```
614E:# OF MEN         2000G (SAM'S VER)
7A5<800.845BM N     3300:# OF MEN
         7FDG     2000G (87 SCTR VER)

BLOAD SEA FOX,A$800 HELLSTORM
6A34:# OF SHIPS      6F25:LV TO START
7A5<800.8960M N 800G  6F4A:# OF SHIPS

FROGGERR        SPACE KADET
70DB:# OF FROGS     5DDE:# OF GUYS
7FDG            7FDG

SUCCESSION      COLOR PLANETOIDS
6B71:# OF GUYS        9B7:# OF SHIPS
6000G           803G

MARS CARS       CEILING ZERO
7024:# OF CARS        356B:09
3FDG            1EC0G (SHORT VER)

NEPTUNE         QUADRANT 6112
8290:# OF SHIPS     980:# OF SHIPS

MARAUDER        RAIDERS / LOST RING
EDIT T1 S3 B46        685A:# OF SHIPS
CHANGE 03 TO 00     803G

LABYRINTH       GALAXIAN
ESC K-A-Y & PRESS   4886:01 TO SET
1-8 TO GO TO THAT   SCORE FOR BONUS
LV OR 9 FOR SHIPS   4800G

CREEPY CORRIDORS    CHOPLIFTER
86A:# OF GUYS         CTRL-L THEN PRESS #
800G            OF LV TO GO TO.

SNAPPER         VIPER
851:# OF GUYS         CCD:C0
7FDG            7FDG

GOLD RUSH       RIBBIT
BE3:# OF GUYS         70DB:# OF FROGS
B00G            6000G

SNEAKERS        BUG ATTACK
6EBB:# OF SHIPS     49D1:# OF BEETLES
C050 C057 329G        8FDG

FALCONS II.......685B:# OF SHIPS
1. RUN GAME 2. REQUEST 1 SHIP
3. HIT RESET 4. 6040G

SPACE QUARKS          BEER RUN
3C54:# OF SHIPS     C64:# OF MEN
BDFG            800G

THRESHOLD -:-
UNLIMITED SHIPS - 45B0:EA EA EA
```

```
              7ECD:EA EA EA
LASER OVERHEAT    - 7666:4C 7D 76
UNLIMITED FUEL    - 7623:EA EA EA
              7839:EA EA EA
TO START GAME     - 6B00G

SNOOGLE - HIT CTRL-SHIFT-M WHEN YOU ARE
A PIE FALLING APART.
SCORING IS AS FOLLOWS:
CHERRY : 100  STRAWBERRY : 300
ORANGE : 500  APPLE      : 700
PLUM   : 1000 BELL       : 2000
GOLDKEY: 3000 KING CROWN : 5000

TAXMAN      (BLOAD)
FOR NO GHOSTS - 505C:EA EA EA
            89CB:# OF GHOSTS
            522B:STARTING LEVEL
            5231:# OF MEN
TO START      - 800G

SWASHBUCKLER          ALIEN AMBUSH
AE0:# OF PIRATES    60E9:# OF SHIPS
1800G           4000G

SNAKE BYTE (BLOAD)
16AE:# OF SNAKES OR 726E:# OF SNAKES
7265:LEVEL OF START
76BD:# OF APPLES TO EAT PER LEVEL
77EAG  OR 250G

GOBBLER         NIGHTCRAWLER
6046:# OF SHIPS     340A:# OF SHIPS
                3300G
```

```
================================================================================
DOCUMENT cheats
================================================================================

1] BLOAD FILE OR BREAK OUT
2] DO MODIFICATIONS
3] TYPE STARTING LOCATION THEN "G" THEN HIT <RETURN>

NOTE:

A "*" NEXT TO THE NAME INDICATES CHEATS THAT NEED AN OLD MONITOR OR
EMULATOR TO USE.

NOTE2:

IF THE STARTING LOCATION IS NOT LISTED OR DOES NOT WORK, TRY TYPING
"AA72.AA73" FROM THE MONITOR. THEN REVERSE AND COMBINE THE BYTES TO
FIND THE STARTING LOCATION.

EXAMPLE:

]BLOAD A.E.
]CALL-151

*F73:EA EA EA

*AA72.AA73

AA72-FD 08

*8FDG

EXTRA CREDITS:

##################
A.E.
F73-EA EA EA
8FD
MAKES BONUS LARGER

A.E.
EE1=# OF MEN
7FD

ALIEN AMBUSH
4608-20 12 46 EA EA
4000
RAPID FIRE

ALIEN AMBUSH
60E9=# OF SHIPS
4000
MAX=80

ALIEN GAME
8550=# OF MEN
800
```

DO THIS BEFORE START (C050 C057 C054)

ALIEN RAIN
4886=# NEEDED FOR FREE SHIP * 1000
11FB

APPLE KONG
43F5-EA EA C050 C057 C054
4000

APPLE PANIC
768F-EA EA EA
4000
UNLIMITED MEN

BEER RUN
C64=# OF MEN
7F8

BELLHOP
6A92=MAXTIP

BERSERKER
SHOOT FROM EDGE OF SCREEN

SHOTS GO THROUGH WALLS
BERSERKER
602F=# OF MEN
1F40

BOLO
14A8-EA EA 1D3B-EA EA
1100
UNLIMITED TANKS

BORG
<SHIFT>-<CTRL>-M-N
SHOWS ALL 10 SCREENS

BUG ATTACK
<CTRL>-B OR <CTRL>-C
B FOR BUG BANISH OR C TO RESTART

BUG ATTACK
49D1=# OF BEETLES
8FD
NO MAX

BUG BATTLE
3FE6-EA EA EA
800

BUZZARD BAIT
8A3F=# OF MEN
2879
MAX=7F

CANNONBALL BLITZ

JUMP AFTER FIRST SCREEN
HALF AS MANY CANNONS ON SECOND SCREEN

CANNONBALL BLITZ
868C=# OF MEN
7FD
MAX=7F

CANNONBALL BLITZ
3C01-EE
7FD
INCREMENTS MEN INSTEAD OF DECREMENT

CANNONBALL BLITZ
608A
START ON SECOND BOARD

CANNONBALL BLITZ
611B
START AT THIRD SCREEN

CANNONBALL BLITZ
6315-60
7FD
CANNONBALL IMMUNITY

CANNONBALL BLITZ
8F77-01
7FD
SPEEDS UP BEGINNING MUSIC

CANYON CLIMBER
3300=# OF CLIMBERS
3000

CHOPLIFTER
E02-20 0F 0E EA EA EA EA
7FD
CONSTANT FIRE

COLOR PLANETOIDS
9B7=# OF SHIPS
803

CONGO
5227-EA EA EA
BEB
UNLIMITED RAFTS

CREEPY CORRIDORS
86A=# OF MEN
7FD
MAX=0F

CRIME WAVE
3BFA=# OF CARS
8FD
NO MAX

CRIME WAVE
1980-EA EA EA 3D89-A9 01
8FD
UNLIMITED SHIELDS

CUBIT
4091=# OF CUBITS
3EFD
NO MAX

CUBIT DELUXE
4097=# OF CUBITS

CYCLOD
8025-EA EA EA
900

DEFENDER
91F-EA EA EA
7FD
INVISO

DIG 'EM
5BB1-4C D1 5B 5BD1-EA EA EA
7FD
UNLIMITED MEN

DIG 'EM
6EAB=# OF MEN
5808
NO MAX

DONKEY KONG
"1" FOR MORE MEN OR "2" FOR INVULNERABILITY

DUNG BEETLES
3D3D-53 CC CF 3D54-CD C2
7FD

ELIMINATOR
17AC=# OF SHIPS
7FD
MAX=7F

EVOLUTION
7904-70 7907-00 6731=# OF MEN
7900
HIT RESET TWICE AT START AND DO MODS

FREEFALL
614E=# OF MEN
7A5<800.845BM N 7FDG
MUST BLOAD AT $800

FROGGER
6504=# OF FROGS
7FD

GOBBLER
6046=# OF GOBBLERS

GOLD RUSH
BE3=# OF MEN
B00

HANDY DANDY
62CD-EA EA
6000
NO WATER

HANDY DANDY
7254-EA EA
6000
NO TIMER

HANDY DANDY
7165-EA EA
6000
UNLIMITED MEN

HARD HAT MACK *
5A2A-60
806
IMMUNE TO CRUNCHERS

HARD HAT MACK *
1660-60
806
NO RIVET

HARD HAT MACK *
4D47-60
806
MUST BREAK OUT AND DO THIS MOD TO ALLOW FURTHER MODS

HARD HAT MACK *
5B80-60
806
IMMUNE TO RIVET

HARD HAT MACK *
A72=# OF MEN
806
MAX=80

HARD HAT MACK *
4DFC-60
806
NO OSHA OR VANDAL

HARD HAT MACK *
95C-EA EA EA
806
CAN'T FALL DOWN HOLES

HARD HAT MACK *
581D-60
806
NO BONUS COUNTDOWN

HARD HAT MACK *
5C40-60
806
IMMUNE TO OSHA AND VANDAL

HELLSTORM
6F25=MAX START LEVEL 6F4A=# OF SHIPS
1200
MAX START LEVEL=80

HORIZON V
5B0A-E6
300
UNLIMITED MEN

HUNGRY BOY
70F4-EA EA
15FD

JAWBREAKER
6046=# OF JAWS
5FFD
MUST BE IN HGR2 TO WORK THEN PRESS <J> OR <K>

JAWBREAKER II*
84B=# OF MEN
800

JELLYFISH
<SHIFT>-2
PRESS AFTER SELECTING CONTROLS FOR A TWO PLAYER GAME

JET PACK
871=# OF MEN
800
MAX=7F

JOUSTER
955-EA EA EA
400<8400.87FFM N 800G
1ST MOD-JOUSTER.2 2ND MOD-JOUSTER.1 AT $8400 FOR UNLIMITED BIRDS

JUMPJET
487B=FUEL
7FD
NO MAX

JUMPJET
116F=ARMS
7FD
NO MAX

JUMPJET

131D-EA EA EA
7FD
UNLIMITED JETS

LABYRINTH
<ESC>-K-A-Y THEN 1 THROUGH 9
1-8 FOR LEVEL OR 9 FOR MORE SHIPS

LODE RUNNER
296F=# OF MEN
800
NO MAX

MARS CARS
7024=# OF CARS
7FB

MILLIPEDE
602A=# OF MEN
1F00

MINER 2049ER
812 AND 814=LEVEL
981
HIT RESET AT 1 OR 2 PLAYER MAX=9

MINER 2049ER
<RESET> AT TITLE PAGE THEN F1C-7 FIBG
SECOND TITLE PAGE

MINER 2049ER
# THEN 1 THROUGH 0
PRESS AT "1 OR 2 PLAYERS" TO START AT ANY LEVEL

MINER 2049ER
816=MEN
981
HIT RESET AT 1 OR 2 PLAYERS MAX=20

MINIT MAN
3F90=# OF MEN
13B3
NO MAX

MONEY MUNCHERS
1020=# OF MEN
FE7
MAX=7F

MOUSKATTACK
6A53-EA EA EA
9FA
ONLY 2 MICE

NEPTUNE
8290=# OF SHIPS
803
MAX=80

NIGHT CRAWLER
340A=# OF SHIPS
3300

NIGHT FLIGHT
8DA9=# OF PLANES
800
NO MAX

NIGHT FLIGHT
A12-EA EA
800
CAN'T DIE

NIGHT MISSION
!
PAUSE

NIGHTMARE GALLERY
1361=# OF GUNS
7F8
MAX=09

NIGHTMARE GALLERY
6818-EA EA EA 8718-EA EA EA 671B-EA EA EA 861B-EA EA EA
7F8

NONADS
41E9-EA EA EA
26FA
UNLIMITED SHIPS

OCEAN KNIGHT
5E06=# OF MEN
800

OUTPOST
2C22 & 8046=# OF MEN
26B0
NO MAX

OUTPOST
3798-EA EA
26B0
NO OVERHEAT

QUADRANT 6112
3E87-EA EA EA
2FC
UNLIMITED MEN

QUADRANT 6112
980=# OF SHIPS
2FC

RAINBOW ZONE
589E=PLANES 581B=LEVEL

801
MAX LEVEL=FE

RASTER BLASTER
692E-EA EA EA EA EA
2700

RASTER BLASTER
8025-EA EA EA
900

REARGUARD
<CTRL>-T
SELECT LEVEL 1-8

REPTON
19C4-4C C8 19 N 19D7-60 N D92-EA EA N
7FD
UNLIMITED MEN AND NUKES

RIBBIT
70DB=# OF FROGS
4B00

RING RAIDERS
685A=# OF SHIPS
47CD

ROBOTRON 2084
40CC-CE 00 00
2DFD
UNLIMITED MEN

ROCKET COMMAND
4563=# OF BASES

SAMMY LIGHTFOOT*
8E00-60
9631
NO BONUS COUNTDOWN

SAMMY LIGHTFOOT*
96BE=SAMMIES
9631
NO MAX

SAMMY LIGHTFOOT*
94E3=SCENE
96C8
MAX=03

SAMMY LIGHTFOOT*
36C=LEVEL
96C8
MAX=0B

SEA DRAGON
8C32-00 8C59-00 8C72-EA EA

5C43
HIT RESET DURING GAME AND DO MODS THAT GIVE UNLIMITED AIR AND DAMAGE

SEAFOX
69D9=MEN
800

SERPENTINE
<ESC>-!-$
FREE SERPENT

SN0GGLE
<SHIFT><CTRL>-M WHEN DYING
3 FREE PUCKMEN

SNACK ATTACK
5B28=# OF MEN
6FD

SNAKE BYTE
76BD=# OF APPLES TO EAT PER LEVEL
77EA

SNAKE BYTE
7265=BOARD
77EA

SNAKE BYTE
16AE AND 762E=# OF SNAKES
77EA

SNAPPER
851=# OF MEN
7FD

SNEAKERS
6EBB=# OF SHIPS
329
MUST START ON HI-RES PAGE (C050 C055)

SPACE KADET
5DDE=# OF SHIPS
7FD
NO MAX

SPACE QUARKS
3C54=# OF SHIPS
BDF
MAX=09

SPARE CHANGE
<CTRL>-Z
ALLOW MODS TO ZERK BEHAVIOR

SPARE CHANGE
<ESC> THEN "-ISLE.DRIVER" FOR A COMPLETE CHEAT MENU

SPY'S DEMISE

60AB=# OF SPIES
800
MAX=80

STAR BLAZER
F69-EA EA EA
300
UNLIMITED SHIPS & FUEL & BOMBS

STAR BLAZER
4800=FUEL 4980=BOMBS 4A80=SHIPS+1
300

STAR BLAZER
131A-EA EA EA
300
SCREWS UP HEAT SEEKING MISSILES

STAR MAZE
459C=# OF SHIPS

STAR MAZE
50B2-EA EA

STAR THIEF
1827=# OF PODS
800
MAX=0B

SUCCESSION
6B71=# OF MEN
7FD

SUPER PUCKMAN
95C-EA EA
800

SUPER PUCKMAN
147B-04
800
ONLY 2 GHOSTS

SUPER PUCKMAN
84D=FRUIT LEVEL
800
MAX=F

SUPER PUCKMAN
1C40-60
800
RUN THROUGH GHOSTS

SUPER PUCKMAN
B82-EA EA
800

SWASHBUCKLER *
AE0=# OF MEN

1800
NO MAX

TALON
<CTRL>-W
ONE FREE MAN (STOP AT 0 - COMPANY LOGO)

TAXMAN
89CB=# OF GHOSTS
800

TAXMAN
505C-EA EA EA
800
NO GHOSTS

TAXMAN
5231=# OF TAXMEN
800

TAXMAN
522B=STARTING BOARD
800

TAXMAN II*
84CA-60
F00
DO MOD THEN "N" THEN F00G

TELEPORT*
41D1-EA EA EA
5F8C

THIEF
4873-EA EA EA
1FF8
LOTS OF MEN

THRESHOLD*
7623-EA EA EA 7839-EA EA EA
6B00
UNLIMITED FUEL

THRESHOLD*
45B0-EA EA EA 7ECD-EA EA EA
6B00
UNLIMITED SHIPS

THRESHOLD*
7666-4C 7D 76
6B00
NO LASER OVERHEAT

TORAX
1E74-A9 00 EA
EF8
RAPID FIRE

```
TUBEWAY
2083-EA EA
7FD
UNLIMITED MEN

TUBEWAY
<ESC>-R-$
ALLOWS SELECTION OF ANY LEVEL UP TO "1-O"

TUBEWAY
22D5=# OF MEN
A00

VIPER
CCD-C0
7FD
UNLIMITED VIPERS

WARGLE
7250=# OF SHIPS

WAVY NAVY
1E63-EA EA EA
931
UNLIMITED MEN

WAVY NAVY
FA7-60 ABC-A0 06
803
HELICOPTERS DON'T SHOOT
---------------------------------------
```

```
==============================================================================
DOCUMENT cheats.app
==============================================================================

[Time:8] Choose (1-11,?,<CR>) 1

*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
*                              *
*      GAME CHEATS - COMPILED BY      *
*                              *
*           THE PENGUIN       *
*           6/13/84           *
*                              *
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

LOST TOMB
---------

BLOAD LOST TOMB
CALL-151
*816D:EA A9 03     (UNLIMITED MEN)
*811B:EA     (UNLIMITED WHIPS)
*7FDG

DRELBS
------

BLOAD DRELBS
CALL-151
*1CB0:EA EA
*C00G

JOUST
-----

BLOAD JOUST (START-UP FILE)
CALL-151
*4B5C:4C 59 FF
*FFDG

CHOOSE OPTIONS YOU WANT AND HIT THE SP.
BAR. JOUST.CODE WILL LOAD AND THEN YOU
WILL BE DUMPED INTO MONITOR. THEN
TYPE:

*B7C3:# OF MEN
*3E00G

ANKH
----

BLOAD ANKH
CALL-151
*6AEB:EA EA
*A00G

GALAXIAN
```

```
--------

BLOAD GALAXIAN
CALL-151
*A83:# OF MEN
*800G

OR FOR INFINITE MEN:

*1751:EA EA
*7FDG

BATTLEZONE
----------

MAXFILES1
BLOAD BATTLEZONE
CALL-151
*98A:# OF TANKS
*810G

BUCK ROGERS
-----------

AFTER STARTING GAME HIT:
CTRL-K
CTRL-E
CTRL-N

POOYAN
------

BLOAD POOYAN
CALL -151
*60ED: # OF MEN
*7FDG

MS. PACMAN
----------

BLOAD MS. PACMAN
CALL-151
*285E: # OF MEN
OR
*27E0:EA EA EA
*17FDG

HEIST
-----

RESET INTO MONITOR FROM TITLE PAGE AND
TYPE:
*F92: # OF MEN
*A00G

OR WITH A SECTOR EDITOR:

T$02
```

```
S$07

EDIT BYTE $92 TO # OF MEN

GUMBALL
-------

WITH A SECTOR EDITOR,EDIT:

T$10
S$0A

BYTES $B8-$BC ARE THE QUOTAS

MR. COOL
--------

BLOAD MR. COOL
CALL-151
*408D: # OF MEN
*4000G

BC'S QUEST FOR TIRES
--------------------

WITH A SECTOR EDITOR READ IN AND EDIT:

T$04
S$0A

EDIT BYTE $11 TO ANY NUMBER BETWEEN
00-80.

ROBOTRON
--------

CTRL-R + (1-99)
EXAMPLE: CTRL-R88 WOULD START AT
LEVEL 88

DIG DUG
-------

BLOAD DIG DUG
CALL-151
*1F6F:69
*1F74:FF
*1F50G

RESET TO MONITOR THEN
*A3DA: # OF MEN
*8000G

RANDAMN
-------

PASSWORDS:
```

```
RISK
TOMB
DROWN
OOZE

DIAMOND MINE
-----------

BLOAD DIAMOND MINE
CALL-151
*1066:5
*7FDG

PASSWORDS:

RKS
QEZ
GEM
WTH

LUNCH TIME
----------

BLOAD LUNCH TIME
CALL-151
*8036: # OF MEN
*4B00G

TALON
-----

CTRL-W (FREE JOUST)

NOTE: I CANNOT GUARANTEE THAT ANY OF
THE ABOVE CHEATS WORK,SINCE I HAVE NOT
TRIED ANY. IF ANYONE FINDS A CHEAT THAT
DOES NOT WORK, PLEASE POST THE CORRECT
INFORMATION.

---------------------------------------
  THE SOUTH POLE ----> [312] 677-7140
---------------------------------------
RAILS WEST! IS COMING SOON! WATCH FOR
      IT AT A BOARD NEAR YOU!
---------------------------------------
      SPECIAL THANKS TO THE WHIP
---------------------------------------
            THE PENGUIN
---------------------------------------
```

```
================================================================================
DOCUMENT cheats2.app
================================================================================

THE SOUTH POLE..........[312] 677-7140

       $$$$$$$$$$$$$$$$$$$$$$$$$
       $ APPLE PIRATE'S CHEATS $
       $$$$$$$$$$$$$$$$$$$$$$$$$

NONADS                BERSERKER
41E9:EA EA EA         6179:EA EA EA
26FAG           1F00G

MONEY MUNCHERS        MILIPEED
1020:# OF MEN         602A:#
FE7G            1F00G

REPTON                REARGUARD
19C4:4C CB 19 N     CTRL-T + LEVEL #
19D7:60 N D92:EA EA

NIGHTMARE GALLERY   APPLE KONG
6818:EA EA EA         43F5:EA EA
8718:EA EA EA         C050 C057 C054
671B:EA EA EA         4000G
861B:EA EA EA

MOUSKATTACK     STAR THIEF
6A53:EA EA EA         1827:# OF SHIPS

EVOLUTION       OUTPOST
6731:# OF GUYS        2C22 & 8046:#
6000G           3798:EA EA

BELLHOP         HORIZON V
6A92:# OF MEN         5B0A:E6 (UNLIM)

STAR MAZE       BLOAD WARGLE.OBJ
50B2:EA EA      7250:# OF GUYS

SUPER PUCKMAN         CONGO
147B:04  1C40:60    5227:EA EA EA
800G            BF4G

CYCLOD                RASTER BLASTER
8025:EA EA EA         692E:EA EA EA EA EA
900G            2700G

DIG 'EM             SEA DRAGON
6EAB:FF         8C32:00 (AIR)
6D87:EA EA EA         8C59:00
5BD1:EA EA EA         8C72:EA EA (DAMAGE)
5808G           5C43G

TELEPORT (RESET)    A.E.
41D1:EA EA EA         EE1:# OF SHIPS
```

```
5F8CG           7FDG

MINER 2049'ER       THE ALIEN GAME
812:# OF LEVEL        8550:#
814:LEVEL - 1         C050 C057 C054
816:# OF MEN          800G
981G

JAWBREAKER II         BOLO
84B:# OF GUYS         14A8:EA EA
800G            1D3D:EA EA

MICROWAVE -:- PRESS RESET ON HIRES PAGE
LESS MONSTERS - 8146:00  8100G
UNLIMITED MEN - EDIT T0 SD B3E
            INSERT EA EA EA
            EDIT T19 SA B3E
            INSERT EA EA EA
UNLIMITED POWER-EDIT T19 SC B75
            INSERT EA EA EA

KAMEARI         SPY'S DEMISE
BLOAD PACK.DATA1    60AB:# OF SPYS
BLOAD PACK.DATA2    3FF1G OR
B82:EA EA   7FDG    C050 C057 C054
95C:EA EA   7FDG    1100G

TUBEWAY         DIG DUG
TRY ESC-R       5BAF:EA EA EA EA EA

SNACK ATTACK          THIEF
5B28:#        6FDG          4873:EA EA EA EA EA

PHASER FIRE     ANOTHER FOR TUBEWAY
452E:# OF SHIPS     22D5:# OF SHIPS
3FFDG           900 OR 7FD OR
                2083:EA EA

STARMAZE (ANOTHER)  SERPENTINE
459C:# OF SHIPS     81A:# OF MEN  7FDG

FREE FALL       CANYON CLIMBER
BLOAD AT A$800        2600:# OF MEN
614E:# OF MEN         2000G (SAM'S VER)
7A5<800.845BM N     3300:# OF MEN
         7FDG     2000G (87 SCTR VER)

BLOAD SEA FOX,A$800 HELLSTORM
6A34:# OF SHIPS      6F25:LV TO START
7A5<800.8960M N 800G  6F4A:# OF SHIPS

FROGGER         SPACE KADET
70DB:# OF FROGS     5DDE:# OF GUYS
7FDG            7FDG

SUCCESSION      COLOR PLANETOIDS
6B71:# OF GUYS        9B7:# OF SHIPS
6000G           803G
```

```
MARS CARS       CEILING ZERO
7024:# OF CARS        356B:09
3FDG            1EC0G (SHORT VER)

NEPTUNE         QUADRANT 6112
8290:# OF SHIPS     980:# OF SHIPS

MARAUDER        RAIDERS / LOST RING
EDIT T1 S3 B46        685A:# OF SHIPS
CHANGE 03 TO 00     803G

LABYRINTH       GALAXIAN
ESC K-A-Y & PRESS   4886:01 TO SET
1-8 TO GO TO THAT   SCORE FOR BONUS
LV OR 9 FOR SHIPS   4800G

CREEPY CORRIDORS    CHOPLIFTER
86A:# OF GUYS         CTRL-L THEN PRESS #
800G            OF LV TO GO TO.

SNAPPER         VIPER
851:# OF GUYS         CCD:C0
7FDG            7FDG

GOLD RUSH       RIBBIT
BE3:# OF GUYS         70DB:# OF FROGS
B00G            6000G

SNEAKERS        BUG ATTACK
6EBB:# OF SHIPS     49D1:# OF BEETLES
C050 C057 329G        8FDG

FALCONS II.......685B:# OF SHIPS
1. RUN GAME 2. REQUEST 1 SHIP
3. HIT RESET 4. 6040G

SPACE QUARKS          BEER RUN
3C54:# OF SHIPS     C64:# OF MEN
BDFG            800G

THRESHOLD -:-
UNLIMITED SHIPS - 45B0:EA EA EA
              7ECD:EA EA EA
LASER OVERHEAT    - 7666:4C 7D 76
UNLIMITED FUEL    - 7623:EA EA EA
              7839:EA EA EA
TO START GAME     - 6B00G

SNOOGLE - HIT CTRL-SHIFT-M WHEN YOU ARE
A PIE FALLING APART.
SCORING IS AS FOLLOWS:
CHERRY : 100  STRAWBERRY : 300
ORANGE : 500  APPLE      : 700
PLUM   : 1000 BELL       : 2000
GOLDKEY: 3000 KING CROWN : 5000

TAXMAN      (BLOAD)
```

```
FOR NO GHOSTS - 505C:EA EA EA
            89CB:# OF GHOSTS
            522B:STARTING LEVEL
            5231:# OF MEN
TO START      - 800G

SWASHBUCKLER          ALIEN AMBUSH
AE0:# OF PIRATES    60E9:# OF SHIPS
1800G           4000G

SNAKE BYTE (BLOAD)
16AE:# OF SNAKES OR 726E:# OF SNAKES
7265:LEVEL OF START
76BD:# OF APPLES TO EAT PER LEVEL
77EAG  OR 250G

GOBBLER         NIGHTCRAWLER
6046:# OF SHIPS     340A:# OF SHIPS
                3300G

CANNONBALL BLITZ - USE INTEGER CARD
AND HIT RESET
868C:# OF GUYS
6147:EA EA EA (PREVENTS CANNONBALLS
             FROM SHOOTING)
608AG FOR 2ND LEVEL START
611BG FOR 3RD LEVEL START

DUNG BEETLES - BLOAD  3D3D:53 CC CF
                  3D54:CD C2
BSAVE DUMB BEETLES,A$7FD,L$4000
```

===
DOCUMENT copyprog.app
===

%%%

How to Copy Programs.
A Beginners Primer.

BY THE THREE MUSKETEERS

%%%

Copy a program is a minor technique easily mastered. The major problem with copying a program is figuring out how it is protected, this is obvious.

To see how a disk is protected, first listen to the drive as it boots up the disk. Be prepared to know what a normal boot sounds like, then check for any differences. If you hear a "swishing" or "syncopated rhythm" the disk is proboably using nibble counting. A procedure in which the number of "nibbles" on a disk is compared to the number actually put on at the company. Strange, as it may seem, a disk with the same information with another disk have differ- ent number of nibbles. If this is found, finding the track is usually easy. It is normally a track that cannot be copied easily for it usually only has a series of one number on the entire track, which nibble copiers tend to be quite irritated at. To copy that track use the option for nibble counting. LS 5.0, EDD 1,2, &3, NA][vA,vB,vC, Copy] all have an option on the menu to "keep" or "preserve" the nibble count. Others like LS 4.1 have parameters to change. (LS 4.1 = 4C=1B).

Another common scheme is to Syncronize the tracks. That is, to place the sectors on one track in a special relationship with another sector on a separate track. The sound of this is an unusually long time on a track. It sounds like
a "swinging pendulum" as it goes from track to track. All copy programs have an option to Sync Tracks. Just choose it.

Other techniques involve changing headers (track starts and data starts) and ending data. Use a "Nibble Editor" to inspect the original disk. You will see, usually plainly, a series of FF's or FE's or some other number (not 96's though...) hese are called Sync Bytes. They tell the program to get ready to receive data. The next bytes are called the header bytes. They tell the computer what track, sector, and volume of the sector. The first three bytes are the start bytes. They tell the computer that this is the Start of Actual Information. Normally they are D5 AA 96. They may be changed. If they are changed, enter the data into the copy program. Usually through parms. although some copiers (one is NA][ ) can enter it from a menu. Later in the data you will see a smaller series of the same Sync Bytes. They are there as a delay. Next comes three more bytes to show that data is next. They are normally D5 AA
AD. If changed, enter the altered bytes into the copier.

These are most of the techinques that are used. But do NOT forget that just a normal run might work.

As homework, try to see the headers in a normal DOS 3.3 disk. Have Fun and Success A.S.R.

==
DOCUMENT copyprot.app
==

#44 : COPY PROTECTION
295 LINES - 59 SECTORS
CONTRIBUTED BY DIAMOND JIM
-----*

COPY PROTECTING YOUR OWN DISKS

BY THOMAS T. BRYLINSKI
08/04/82

INTRODUCTION:

For those new-commers to the world of APPLE Computers, and to the history of software development, here is a brief summary. In ancient times (1978-1979), the APPLE Corporation was just getting started , and absolutely no software was available for your $1530 toy. So most people who bought this expensive little tan box had to write their own software. If you were among the more fortunate users who had a good sales pitch, you talked your boss into buying you an APPLE, and then spent your company's time learning the in's and out's of programming. In any case, you could not purchase ANY good software for your mac hine. Shortly after the first early programmers crawled out of their shells, APPLE users groups started to form. The prime function of these groups was to share programs and to exchange the secrets which one had learned in the previous month.   (also it was a good excuse to get away from the kids at home, for a night)! Four or five months passed and a few early programmers got the idea that they would market their software and make a few bucks for their hard hours programming. And thus, the first APPLE soft ware companies formed. These companies were very small and usually started in someone's basement. The prime buyers of this software were the APPLE DEALERS. The dealers could now demonstrate these marvelous machines with some "GREAT" software. By the way, this great software came on cassettes, (you know, those little plastic things you used to record music on). These cassettes were copyable by normal means, (eg. tape recording), and the dealers started giving some programs away with each system that theys old.

In the summer of the DARK AGES (1979), APPLE COMUPTER released their first d isk drive system (3.2 DOS). This disk system made copying programs easier, faster, and much more reliable. At this time copying was encouraged by both programmer and dealer. So on the software companies realized the increasing market for their products, and theorized that if they could produce a disk that could not be copied by normal means they could sell more software, hence more profit. APPLE'S disk system was the perfect answer to their problem. APPLE chose to make their disk system totally "SOFT", which means that all information pertaining to the disk operation is stored on disk . This information is then loaded into RAM (random access memory), upon a system boot (PR#6). All commands typed at the keyboard are examined by the "disk operating system" (DOS), and then by the apple ROM's (that row of big fat chips inside the machine). Now the software writers had an edge on the normal user, change how the APPLE responds to user commands, and keep them out of your programs. The only problem was that the copy program that came with your disk drive was able to copy the complete disk. With a "soft" DOS, the programmers could change how the information is read from disk and modify their DOS to read it. As Apple users became more aware of the internal workings of their machines, programmers made more and more changes to DOS, and the race was on!

So much for the history lesson (boring out-of-date information anyway), and o
n to the meat of the lecture.

TERMS USED IN THE TEXT:

BIT-	the smallest piece of information that the computer can recognize or process.
NIBBLE-	four bits in a row, or a block
BYTE-	eight bits in a row or block. It is the smallest piece of information that people like to work with. (00000000)
VTOC-	Volume Table Of Contents: DOS uses this sector to tell it which sectors are used and which are free on the disk.
SELF-SYNC BYTE-	a special byte used for locating information on the disk. This byte differs from a normal byte in that it is made up of nine bits. (111111111)

PROTECTION METHODS

DISK COMMAND CHANGES- changes to the DOS that make those familiar words like Load, Delete, and Save, give the user that cold, unforgiving response...SYNTAX ERROR

CATALOG TRACK LOCATION- moving the catalog to a non-standard track (normally track HEX $11, DEC 17)

CHECKSUM ALTERATION- the portion of each sector that DOS automatically checks to make sure that the information it has read is correct.

$D6,VECTOR- an Applesoft pointer used by the machine to make "carriage return" = RUN.

LOADER DOS- a DOS whose sole purpose is to Load and execute one program from disk.

HALF-TRACKING- writing information between the normal tracks on the disk.

DOS HOOK- designating a specific track on the disk, where the only information on the track is a track & sector number, to tell DOS where to read next.

PROGRAM LOCK- a line of programming that looks at a specific memory location and compares its contents to a programmed number. (x=peek(y))

HARDWARE LOCK- Using a hardware modification to lock the program.

NIBBLE COUNTING- setting aside a specific track on the disk where a number of self-sync nibbles are written.

TOOLS FOR LOCKING PROGRAMS

DOS BOSS - Beagle Brothers Software
BEANETH APPLE DOS - Quality Software
PROGRAMMER'S AIDS - Dakin 5 Corporation
BAG OF TRICKS - Quality Software
SUPER DISK COPY - Sensible Software
TASC - Microsoft
THE EXPEDITER - On Line Systems

THE DOS MANUAL - Apple Computer Corporation
APPLE II REFERENCE MANUAL - Apple Computer Corporation
WHAT'S WHERE IN THE APPLE - William F. Luebbert
SOFTALK magazine
NIBBLE magazine

If you are familiar with the above manuals, software, and periodicals you ar e well on your way to locking programs. Also you will need use of one of the nib ble copiers on the market such as, LOCKSMITH, NIBBLES AWAY, or CLONE. CLONE is my choice because it is very fast compared to the others.

LOCKING TECHNIQUES:

MESS UP DOS

Change some or all of the DOS commands. This in itself may be enough to prot ect your programs. Go a little further. Bury some control characters in the cata log. (control chrs. don't print usually). Change "CATALOG" to "LIST" and the Bas ic command "List " becomes unusable. Try it, you can't "list" a program in memory. Duplicate DOS commands are great. Only the first one encountered will work. Confuse the user by changing the disk error messages. For example do the following:

1) Change the SAVE command to STORE
2) Change the READ command to SAVE
3) Change the "NOT DIRECT COMMAND" error
   message to "NOT COPYABLE"

Now when anyone tries to load and save your program you get the "NOT COPYABLE" error message because he used the wrong command! *** EXPERIMENT ***

Now the following can be done to any disk you want. We will move the catalog track from track $11 to track $5, just for convenience mind you.

1) Boot your favorite 3.3 system master to load DOS
2) Placeyour DOS BOSS disk in the drive and type:LOAD DOS BOSS (return)
3) Type: Poke 44033,5 (return)
4) Place a blank disk into the disk drive and close the door. (something your
   parents keep telling you to do.)
5) Type: RUN (return)
6) Change a few commands...any one you want!
7) Before you leave DOS BOSS, change the disk volume heading to" SYNTAX ERROR"
... Don't forget the ctrl-G at the end!
8) Exit the DOS BOSS program.
9) Type: NEW (return) <--(by now you should remember)
10) Type: INIT HELLO
11) Wait a minute or so and pull the disk out of the drive.
12) Boot your system master again and try to catalog the disk you've just init
    iallized.

If you have not noticed by now 44033 is the memory location that holds the cata log track number. Type: PRINT PEEK(44033), and you will see that DOS is looking at track 17 to find the catalog. Now if one were rather clever you would use som ething like SUPE R DISK COPY to copy the catalog track from another disk onto your modified disk . Also it will be necessary to change VTOC so that you do not overwrite real fil es on the disk. VTOC is normally located on track $11, sector $00. However the V TOC to fix on yo ur modified disk i **

```
   T0 SYNC: 18=20 19=00 40=20 44=DD 45=AD
          46=DA 72=00 73=00 77=00
          78=00 79=12 7C=00

   T1.5-TB.5  SYNC
   TD-T20  SYNC

  BORG **

   T0: 18=20 19=00 40=20 4D=00 4E=00
       52=00 53=00 54=12 57=00
       72=00 73=00 77=00 78=00 79=12
       7C=00 44=DD 45=AD 46=DA

   T1.5-TC.5 SYNC
   TD-T20 SYNC
```

BPI BUSINESS ACCTING SYSTEM (4 DISKS)

(REVISED 10-26)

```
  T0-T22: 19=00 21=02 58=19 59=06        5A=1A 5B=FF BD=44 BE=E6
  BF=45 C0=FF C1=40 C2=01

         C4=44 program RUN when any command is issued.
```

POKE 1010,102: POKE 1011,213: POKE 1012,112 -- Makes RESET run the program in m emory.

POKE 2049,1 -- Makes the first program line list repeately.

Well by this time you should be bored stiff or really into learning copy pro tection. If the latter is the case continue to read, if the former, re-boot the system and fire up your favorite game.

Now we shall take on the heavier ways to protect. If you were reading carefu lly to this point, you now should know how to change your DOS commands and chang e the catalog track. Also if you were experimenting you should have a few other tricks under you r belt. So, if you're having trouble at this point it would be advised to start at the beginning!

In this section we will discuss the heavier ways of protection.

CHECKSUM ALTERATION:

In each sector on the disk is a byte which is the Checksum. This byte is the last byte to be written into a sector.   The value of this byte varies with the a mount of information stored in that sector. Normal Apple DOS reads in the inform ation on the sec tor, and then counts the bytes it has read. It then compares this number to the checksum, if they are equal it continues to read the next sector. If it is not equal DOS has made an error and tries to read it again.   After three tries it sto ps and gives the user an error message. In order to change the checksum we must change the byte should also be noted at this time, that your standard 3.3 DOS will no longer re ad this sector.

Now in order to read this sector, we must disable the Checksum routine in DOS. To do this from the keyboard type the following:

```
  1) CALL-151
  2) B942:18       REM 3.3 DOS
       or
     B963:18       REM 3.2 DOS
```

This changes a "set carry" instruction to a "clear carry" instruction.

```
  3) 3D0G
```

Now you're back in Basic.

I hav'nt found a way to INIT a disk with this changed DOS yet, but by using DAK IN 5 PROGRAMMERS AIDS you can change DOS directly on your disk with the Patcher. The data to be zapped resides on track 0, sector 3.

```
    Byte $42  change $38 to $18  REM 3.3 DOS

    Byte $63  change $38 to $18  REM 3.2 DOS
```

$D6, VECTOR:

The D6 memory location in the Apple can set from Applesoft by typing POKE 21 4,255; OR from assembly by:

```
           LDA  #$FF
           STA  D6
```

This is where the Applesoft Run pointer resides. By putting a number larger tha n 128 in this location Applesoft equates a carriage return with the Applesoft RU N command. Once set, all user commands cause the program in memory to be execute d.

LOADER DOS:

Loader DOS is the minimal DOS that can be utilized in the Apple. It consists of nothing more than RWTS and a table of track and sector numbers that are to b e read in. Loader DOS has no DOS commands, as its only function is to load a pro gram, and start running it. If you're interested in this consult the DOS manual. The manual exp lains how to write the look-up table and how to utilize RWTS directly.

HALF-TRACKING:

half- tracking is utilizing the tracks between the normal tracks on the disk . This is possible because the disk drive is actually capable of writing to seve nty tracks, as that is the number of stepped positions the read/write head has. However one cann ot use these half tracks to double the amount of information stored on the disk due to hardware constraints in the Apple drive unit. In order to use half track s the adjacent full tracks must not be written to because of the high risk of ov erwriting or des troying information on the half track.   It is only possible to write to half tra cks with assembly because the programmer must toggle the soft stepper switch onl y once and then access RWTS directly.

DOS HOOK:

In order to use a DOS HOOK one has to first write their own RWTS portion of DOS. Then write or modify the DOS boot routines to supply RWTS with a track and sector number and read that sector. This information is taken as data for RWTS a nd the next read . A program that utilizes the hook very effectively is MASTERTYPE from Lightning Software.

PROGRAM LOCK:

This is no more than a combination lock that is built into the program. To e ffectively use it, it is necessary to modify the boot routine in DOS. This is done by moving the PROM boot routine down into RAM where we can change it to stop after the first bootstrap routine is loaded. This is done by typing:

```
 1) CALL-151
 2) 9600<C600.C700M
 3) 96F9:59 FF
 4) 9600G
```

At this point the disk starts and loads the boot routine in at $800 but does not execute it. Now look at it by typing 800L. Hit L a few more times until you come to JMP $301. The OP codes should be 4C 01 03. This is the key that you will look for on the d isk. You will find them on track 00, sector 00 of the disk. Using PROGRAMMING A IDS you will be able to change this information on the disk, and put into memory your own combination. Do this by typing in the
OP codes for the following:

```
       LDA  #$XX      ;XX = PART OF COMB

       STA  YYYY      ;YYYY = MEMORY LOC
```

And don't forget to put the JMP $300 back in.

Now all that is left is to doctor up your program to look for the combinatio n that you stored in the boot. Do this by PEEKing that memory location, and comp aring the contents.

HARDWARE LOCK:

I won't spend much time on this because it is the worst way to protect softw are. It works like this: You have to plug in something that looks like an integrated circuit into the game port. That will simulate the game paddles set at a specific spot. The program then reads the port and compares the input to the progeammed readings , if different....CRASH!!!

NIBBLE COUNTING:

Unfortunately the only thing I know for sure about this is it must access the memory locations C080-C08F+16*(SLOT #)

SUMMARY:

If you choose to write your programs in Basic, it is a very good idea to comp ile the source code. The generated OP CODE is almost impossible to read or change. In this way you can hide all sorts of locking schemes. Also don't forget to use the ONERR Applesoft command, this will stop a ctrl-C Break from Applesoft.

-----*

```
==============================================================================
DOCUMENT correct.app
==============================================================================
```

Advanced Programming Information Fixed

30-MAR-85 By Homer Brothers Software

Please upload this doc on every bbs you use so that everyone will start programming your Cat correctly.

Along time ago when the Novation Apple Cat was first released, Novation thought that they would be nice people and give all you hackers information on how to diddle with there hardware. Well since that time many of us have found that they messed up in a few places, in fact, Advanced Programming Information has more bugs than COM-WARE did. To my knowledge the API manual has never been de-bugged.

So please throw away your programming ego for a few minutes and take the time to read this file.

Thanks
Homer Brothers

P.S.   I never want to see another 212 card turn on unless it was suppose to!

The most important bug in the API manual is the 212 card bug. Please turn to page 7 students. If you will notice the SQUBYT register's hi order bit is the 212 disable/enable bit. Yes dreaded ol' bit number 7 must be on to disable the 212 card just like it says here. That means to have the handset squelched the cassette off and the 212 card off, you would need to store a $81 in SQUBYT. Now your saying well thats not a bug in the manual, well your correct, that actual bug that has confused so many is on another page. Please turn to page 24 students. Please notice the modem INIT routine where they load the accum with binary 00000001, a hex 01... Right about now your saying damn I shouldn't have cluged that code I knew it sucked the second I looked at it. Well give him a break, he wrote this thing probably before the 212 card was finished.

Ok class, you have learned of the most common error in programming the Cat. Well now, if all you stud programmers have managed to stay with us, let me please point out some of the more obscure mistakes in programming the cat. (Ones even total stud programmers have made)

Please turn to page 15 students. Ah yes, the dreaded XMTBYT. The cause of so many early apple-cat repairs. Yes believe it or not, many programmers never bother to shut off the carrier when they hang up the phone line (As the micron did in his Catsend bbs). Leaving on the carrier after hanging up the phone is not good for the poor little heat sensitive LSI chips that Novation designed, so you can only add to the life of them by giving the carrier a rest after the caller is logged of by powering there bod's down. That means a $1F to the XMTBYT and a extended life time for the Cat.

Ok, well so what that wasn't a bug in the manual heres another for you.

Please turn to page 4 students. Ah yes the much loved SWBYT. This do all register has been so misunderstood because of the mistakes in API. Ah those lovely firmware switches, when used with the firmware protocol they can tell you your modem defaults, ie 212 installed?   Welp as you may have already noticed they botched the bit order on the switches. The register should read-

```
           SWBYT

7   6 5   4 3   2 1   0
DV  BSR CD  AD    SW3 SW2 SW1 SW4
```

Ok well you thought I would never find more than one bug... So whats next?

Well students flip to page 21. On the bottom of the page you should see the Label CHKRNG, as you can see they check ACBYT's ring bit. If the phone is ringing, they print a nice little message and wait for the line to stop ringing. Now if your saying, well thats stupid, why don't they just pickup the phone?, well that would be okay with most modems. BUT! Novation botched the hardware on the Cat, and if you pick up the line while the 55 volt ring pulse is there, WHAM! the cat takes a jolt that nocks its registers silly.

Well by now your saying, thats it, nothing else... Well I am not out of breath yet.

Bad programmers (shame on you) do somthing like this when they init there hardware.

```
INIT   SEI
       LDY SLOT
       LDA #%10000001
       STA .... etc etc etc
       init init init....
       RTS
```

More experienced programmers (I love you guys) do somthing like this.

```
INIT   PHP
       SEI
       LDY SLOT
       LDA #%10000001
       STA .... etc etc etc
       init init init....
       PLP
       RTS
```

Hey, he knows that other devices use interupts besides the CAT... boy that guy must use ProDOS.

Well thats realy about all the bad things I can say right now. I will try to come up with some more. Please take the time to make sure you understand what I did here if you plan on programming your cat from 6502 machine code, then rip it up and say you knew that a lot longer than Homer did.

Boy don't we programmers have big easily dented ego's? I know I do.

OH! I cant end this file without saying this.

REAL MEN USE EDASM ProDOS! Boys play with Merlin and its wimpy little symbol tables. Oh yeah and, REAL MEN USE A DCI BASED PRINT

Whats that? This-

```
*
* PRINT, DCI based of course
* By Homer Brothers, some time in
* the late 80's
*

TINDR0        EQU $E0
        LDA #0
        STA TINDR0

* Above only need be done once in the
* begining of your program.

LOOP    JSR PRINT
        DCI "How the hell are you? "
        JMP LOOP
PRINT   EQU *
        PLA
        TAY
        PLA
        STA TINDR0+1
PRINTLOOP INY
        BNE GETNCHAR
        INC TINDR0+1
GETNCHAR  LDA (TINDR0),Y
        PHP
        ORA #$80
        JSR COUT
        PLP
        BPL PRINTLOOP
        LDA TINDR0+1
        PHA
        TYA
        PHA
        RTS
```

  Make sure that when you use this, you tell everyone that you used it long before Homer did, or that you would have thought of it anyways.

  God would somone please beat my ego up please.

Homer Brothers
(312) 665-0264

```
=============================================================================
DOCUMENT cr.adder
=============================================================================
```

Larry Sholl
76324,1413
K1EPC
903 Walton St.
Rockville, In.
July 14, 1987

Carriage Return Adding

AppleWorks data bases may be created from text files if each "category" entry is demarcated with a carriage return. I have found this feature to be very useful when converting a text listing of names and addresses to a data based format. Unfortunately, text downloads don't contain the needed CR's and it is a pain to use a word processor to format a large file.

CR.ADDER is a slow basic program which will replace the second of two consecutive spaces within a text file with a CR. Usually, the original text file needs dressing up on the word processor to assure that the start of the file will contain the first category entry. Print, using your word processor, the file to disk as a text file. Run CR.ADDER and a modified file will be saved as name.CRA. AppleWorks can now be used to create a database from this file if you correctly definine the number of categories per record.

```
==============================================================================
DOCUMENT crack1.txt
==============================================================================
```

I N T R O D U C T I O N

 THE SUBJECT OF SOFTWARE PIRACY HAS BEEN AN OPEN TOPIC OF DISCUSSION ON THE BOARD FOR SOME PERIOD OF TIME. IN THE MONTH OF NOVEMBER A PERSON CAME ON OUR BOARD WHO CALLED HIMSELF ROBBING HOOD, WHO SAID HE LIVED AT RISLEY HALL AT CORNELL, WAS A SOPHOMORE AND ALSO HIS REAL NAME WAS FRED WILLIAMS. IN EARLY DECEMBER I FOUND OUT THAT HE WAS REALLY SOMEONE FROM THE COMPANY CALLED SIRTECH WHICH PRODUCES A GAME CALLED CALLED WIZARDRY. SOMETIME IN JANUARY HE FOUND OUT THAT WE KNEW WHO HE WAS AND AFTER SOME DISCUSSION DECIDED TO 'COME CLEAN'. I TOLD HIM THAT HE COULD TELL HIS STORY AND THAT WE WOULD HAVE AN OPEN DISCUSSION OF THE SUBJECT. COMMENTS HAVE BEEN MADE BY MANY USERS, AUTHORS, ATTORNEYS, PIRATES AND CRACKERS. THE ONGOING DIALOGUE IS GIVEN IN THE FOLLOWING VOLUMES OF THE SUBJECT.

MAKE UP YOUR OWN MIND

NOTE: THE FOLLOWING FILES ARE AS LONG AS 100 SECTORS EACH. PLEASE USE A
      DATA CAPTURE TO SAVE THESE FILES.

```
================================================================================
DOCUMENT crackdos.app
================================================================================
```

```
:::::::::::::::::::::::::::::::::::::::
::*:::::::::::::::::::::::::::::::::*::
:::                            :::
:::   DISK TIPS & TRICKS TUTORIAL   :::
:::         PART 1                :::
:::     INTRODUCTION TO DOS       :::
:::                            :::
:::       BY CANDY APPLE          :::
:::                            :::
::*:::::::::::::::::::::::::::::::::*::
:::::::::::::::::::::::::::::::::::::::

   *:::::::::::::::::::::::::::::::::*
   ::     DISTRIBUTED BY        ::
   :: CRYSTAL CASTLE 313/856-3804 ::
   *:::::::::::::::::::::::::::::::::*
```

THIS ARTICLE CONTAINS AN OVERVIEW OF HOW A NORMAL DISK IS FORMATTED AND DISCUSSES BRIEFLY THE TWO WAYS OF READING A DISK. IT IS WRITTEN FOR THOSE WHO AREN'T FAMILIAR WITH DOS OR FOR THOSE WHO WANT A SHORT REVIEW ON DISK FORMATTING, ETC.

FIRST LET'S START OFF WITH A LITTLE BACKGROUND ON HOW DOS FORMATS A DISK. WHEN A DISK IS 'INIT'IALIZED BY DOS, IT WILL DIVIDE IT INTO 35 CONCENTRIC TRACKS. IF YOU HAVE DOS 3.3, THEN EACH TRACK WILL BE DIVIDED INTO 16 BLOCKS CALLED SECTORS, WHEREAS DOS 3.2 WILL CREATE 13 SECTORS. ON EACH OF THESE SECTORS ARE AN ADDRESS MARK AND A DATA MARK.

THE ADDRESS MARK WILL TELL DOS WHAT TRACK AND SECTOR IT IS CURRENTLY READING. WITHIN THIS ADDRESS MARK, THE VOL- UME, TRACK, SECTOR, AND CHECKSUM INFORMATION CAN BE FOUND. THE DATA MARKS SURROUNDS THE ACTUAL DATA AND TELLS DOS WHERE THE DATA BEGINS AND ENDS. IT ALSO CONTAINS A CHECKSUM THAT'S USED TO VERIFY THE ACCURACY OF THE DATA.

THE TRACKS ARE NUMBERED FROM $00 (0- DEC.) TO $22 (34-DEC.); WHEREAS THE SECTORS ARE NUMBERED FROM $00 (0) TO $0F (15). THE DOS PROGRAM USES TRACKS $00 THRU TRACKS $02 (A TOTAL OF 3 TRACKS;0,1,2).

THE DOS ALLOWS THE APPLE TO MANIPULATE DATA ON A DISKETTE. WITHIN THE DOS PRO- GRAM ARE ALL OF THE COMMANDS THAT CON- TROL THE DISKDRIVE (I.E.: CATALOG, INIT, LOAD...) AND THE ERROR MESSAGES WHICH YOU HAVE PROBABLY SEEN BY NOW.

ON THE DISK CONTROLLER CARD THAT CON- NECTS THE APPLE TO YOUR DISK DRIVE IS A SMALL PROGRAM, SO THAT WHEN YOU BOOT A DISK, IT WILL TELL THE DISK DRIVE TO READ TRACK $00(0). THE PROGRAM ON TRK $00, SCT $00 CONTAINS THE INFORMATION TO READ IN SECTORS $00-$09 ON TRACK $00. THIS PROGRAM ON SECTORS $00-$09 WILL READ IN THE REMAINING INFORMATION ON TRK $00-$02. IN OTHER WORDS, THE APPLE HAS NOW LOADED THE DOS, AND DOS WILL NOW TAKE OVER AND RUN THE PROGRAM YOU HAVE INITIALIZED THE DISK WITH.

DOS FINDS THE HELLO PROGRAM BY GOING TO THE VOLUME TABLE OF CONTENTS (VTOC) AND DIRECTORY THAT'S LOCATED ON TRACK $11 (17). THE VTOC OR "BIT MAP" WILL SHOW WHICH SECTORS ARE USED AND WHICH ARE FREE. THE DIRECTORY BEGINS ON SECTOR

$0F(15) AND CONTINUES ON DOWN TO SECTOR $01(19). THE VTOC AND DIRCECTORY ARE USED BY DOS WHEN YOU SAVE OR DELETE A FILE.

WITHIN THE DIRECTORY, YOU WILL FIND A LIST OF ALL THE FILES FOUND ON THE DISK. EACH ENTRY CONTAINS A POINTER TO THE TRACK/SECTOR LIST, FILE-LOCKED AND FILE-TYPE CODE, FILE-NAME AND FILE-SIZE WHICH WE'LL GO INTO MORE DETAIL LATER. ACTUALLY THE TRK/SCT LIST IS A LIST OF THE TRK/SCT PAIRS THAT STORE THAT FILE. THAT'S WHY SAVING A BLANK FILE ALWAYS TAKES 2 SECTORS BECAUSE ONE IS USED FOR THE BLANK FILE AND ONE IS FOR THE TRK/ SCT LIST.

BECAUSE THE CATALOG TRACK IS IN THE CENTER OF THE DISK, THE ARM NEVER HAS TO TRAVEL MORE THEN 17 TRACKS TO GET TO THE CATALOG TRACK. THEN AS FILES ARE ALLOCATED ON THE DISK, THEY WILL OCCUPY THE TRACKS JUST ABOVE THE CATALOG FIRST UNTIL IT REACHES TR $22(34), THEN IT WILL START USING THE TRACKS JUST BELOW THE CATALOG $10(16), THEN TRACK $0F(15) ETC, MOVING TOWARDS THE DOS IMAGE TRACKS.

ANOTHER THING THAT'S INTERESTING TO KNOW IS THAT WHEN YOU 'SAVE' OR 'BSAVE' A PROGRAM, DOS WILL TOTALLY ALTER THE PROGRAM CODE BEFORE IT IS WRITTEN TO THE DISK. THEN WHEN YOU 'LOAD' OR 'RUN' THAT PROGRAM, DOS WILL CHANGE IT BACK AGAIN TO ITS ORIGINAL FORM, ALTHOUGH THE USER NEVER NOTICES THIS PROCESS.

YOU MIGHT WONDER WHY DOS GOES TO ALL THIS EXTRA TROUBLE? ACTUALLY THE AP- PLE'S HARDWARE (AND THAT OF OTHER MA- CHINES) HAS SOME LIMITATIONS WHICH RE- STRICT THE RANGE OF BYTE VALUES THAT CAN BE ALLOWED TO PASS BETWEEN THE MA- CHINE AND ITS DISK DRIVES.

IF YOU TYPE "CALL -151" AND LIST A RANGE OF MEMORY (SAY 'F800.FFFF'), YOU WILL NOTICE THAT ALMOSE EVERY BYTE VALUE FROM $00 TO $FF CAN BE SEEN SCROLLING BY. EVERYTHING THAT'S STORED IN RAM SUCH AS PROGRAMS, TEXT FILES, ETC.., IS REPRESENTED BY A BLOCK OF HEX VALUES IN THIS RANGE. THIS MEANS THAT THE APPLE HAS 256 DIFFERENT BYTE VALUES TO USE FOR REPRESENTING INFOR- MATION IN MEMORY (THERE ARE 256 DIFFER- ENT HEX NUMBERS IN THE RANGE OF $00 TO $FF).

UNFORTUNATELY, WHEN THE APPLE COMMUNI- CATES WITH THE DISK DRIVES, IT CAN'T HANDLE SUCH A LARGE RANGE OF VALUES DUE TO HARDWARE CONSTRAINTS. DOS 3.3 CAN ONLY SEND TO THE DISK OR RECIEVE FROM IT THE VALUES $96 TO $FF (150 TO 256). EVEN SOME OF THESE BYTES WITHIN THAT RANGE ARE ILLEGAL BECAUSE THEY VIOLATE APPLE HARDWARE RULES, AND OTHERS ARE RESERVED FOR SPECIAL DISK USE. ACTUALLY DOS 3.3 HAS TO REPRESENT ALL 256 DIF- FERENT VALUES THAT APPEAR IN RAM USING ONLY 64 VALUES ON THE DISK. EARLIER VERSIONS OF DOS HAD TO MAKE DO WITH A SMALLER RANGE OF DISK BYTES.

THERE ARE 2 WAYS OF READING A DISK:

1) A RAW NIBBLE DUMP
2) AN RWTS READ.

"RAW NIBBLES" REFERS TO INFORMATION EX- ACTLY AS IT IS REPRESENTED ON DISK - IN THE SPECIALLY ENCODED FORM DESCRIBED ABOVE.

NIBBLE DUMP YOU WILL NOTICE (ON A DOS 3.3 DISK) THAT THE HEX NUMBERS WILL BE BETWEEN THE VALUES $96 TO $FF.   THIS IS HARDLY RECOGNIZABLE AS PROGRAM OR TEXT FILE CODE. YOU WILL ALSO SEE DOZENS OF HEX NUMBERS WHICH HELP DOS DO ITS JOB IN GETTING INFORMATION ON AND OFF THE DISK. THESE "DOS MARKS" WILL TELL YOU A GREAT DEAL ABOUT THE DISK AND WILL BE COVERED IN A LATER TUTORIAL.

THE OTHER WAY A DISK IS READ IS THROUGH A SUBROUTINE IN DOS CALLED "READ AND WRITE TRACKS AND SECTORS" OR OTHERWISE KNOWN AS RWTS. THIS ROUTINE PUTS DATA

(WRITES) ON THE DISK AND GETS IT BACK (READS). ONE OF ITS IMPORTANT JOBS IS TO TRANSLATE THE RAW NIBBLES FROM THE DISK INTO INTELLIGIBLE CODE FOR APPLE ROMS AND PROGRAMMERS.

THE RWTS WILL ALSO FILTER OUT THE DOS MARKS REFEREED TO EARLIER, BECAUSE ONCE THE PROGRAM IS LOADED INTO MEMORY, THE DATA MARKS SERVE NO PURPOSE AND SO THEY ARE DISCARDED AFTER BEING PICKED UP BY THE READ/WRITE HEAD. JUST REMEMBER THAT WHEN DOS LOADS AND RUNS ANY PROGRAM, IT WILL AUTOMATICALLY PERFORM AN RWTS READ TO GET THE PROGRAM INTO MEMORY, WHEREAS A RAW NIBBLE DUMP COMES DIRECTLY FROM THE DISK AND BYPASSES RWTS AND DOS ENTIRELY.

IN THE NEXT TUTORIAL, WE'LL GET DOWN TO BUSINESS AND START EXAMINING THE VTOC AND ITS INNER WORKINGS.

INCLUDED BELOW IS A HEX, BINARY, DECI- MAL CHART WHICH MIGHT PROVE USEFUL TO YOU IN LATER TUTORIALS.

HEX	BINARY	DECIMAL
0	0000 0000	0
1	0000 0001	1
2	0000 0010	2
3	0000 0011	3
4	0000 0100	4
5	0000 0101	5
6	0000 0110	6
7	0000 0111	7
8	0000 1000	8
9	0000 1001	9
A	0000 1010	10
B	0000 1011	11
C	0000 1100	12
D	0000 1101	13
E	0000 1110	14
F	0000 1111	15
10	0001 0000	16
20	0010 0000	32
3F	0011 1111	63
40	0100 0000	64
7F	0111 1111	127
80	1000 0000	128
AA	1010 1010	170
C0	1100 0000	192
E8	1110 1000	232
FE	1111 1110	254
FF	1111 1111	255

```
=======================================
:::::::::::::::::::::::::::::::::::::::
::*:::::::::::::::::::::::::::::::::*::
:::                        :::
:::   DISK TIPS & TRICKS TUTORIAL   :::
:::         PART 2               :::
:::   CLOSER LOOK AT THE VTOC    :::
:::                        :::
:::       BY CANDY APPLE         :::
:::                        :::
::*:::::::::::::::::::::::::::::::::*::
:::::::::::::::::::::::::::::::::::::::
   *:::::::::::::::::::::::::::::::*
   ::     DISTRIBUTED BY          ::
   :: CRYSTAL CASTLE 313/856-3804 ::
   *:::::::::::::::::::::::::::::::*
```

THIS TUTORIAL WILL REPRESENT THE FIRST LEG OF YOUR JOURNEY TOWARDS DISK EXPERTISE. LET IT BE KNOW N THAT THE DOS MOD- IFICATIONS EXPLAINED BELOW AND HEREAFTE R ONLY APPLY TO A STANDARD DOS 3.3 SLAVE DISK UNLESS OTHE RWISE STATED. A DISK WITH ANY OTHER TYPE OF DOS SUCH AS DOS 3.2 OR A FAST DOS WILL HAVE LOCATIONS WHERE MANY OF T HESE PATCHES WON'T WORK AND COULD EVEN RUIN A DISK. IN FA CT IT'S BEST TO MAKE THESE PATCHES ON A SCRATCH DISK UN TIL THINGS WORK PROPERLY. THE AUTHOR IS NOT RESPONSIBL E FOR DAMAGE DONE TO ANY DISKETTE.

TO CONTINUE WITH THIS TUTORIAL, YOU WIL L NEED A DOS 3.3 SLAVE DISK AND A TRACK AND SECTOR EDITOR PROG RAM SUCH AS DISK EDIT, THE INSPECTOR, TRICKY DICK, OR THE SECT OR EDITOR ON COPY II+, ETC. ALSO, SO WE CAN WORK AND UNDERSTA ND ON THE SAME LEVEL, MAKE A 'FID' COPY OF THE APPLE SYSTEM M ASTER DISKETTE AND LEAVE IT UNPROTECTED. NOW LETS GET STA RTED.

THE VOLUME TABLE OF CONTENTS
--- ------ ----- -- --------

USING YOUR DISK EDIT PROGRAM, READ IN T RACK $11(17), SECTOR $00(0). YOU ARE NOW LOOKING AT THE VTO C WHICH WILL SOONER OR LATER IN YOUR LIFE GET OVERWRITTEN WITH WORRY ABOUT FIXING A CLOBBERED VTOC RIG HT NOW BUT IN A LATER TUTORIAL WE'LL DISCUSS THIS.

LET'S TAKE A CLOSER LOOK AT THE FIRST L INE OF DATA STARTING WITH BYTE $00. IT SHOULD LOOK SIMILIAR TO THE FOLLOWING:

```
BYTE  TRACK $11   SECTOR$00

00:  04 11 0F 03 00 00 FE 00
```

THE FIRST NUMBER "04" DOESN'T HAVE ANY FUNCTIONS, SO IGNORE IT. THE NEXT TWO NUMBERS GIVE US THE F IRST CATALOG ENTRIES. THE SECOND BYTE REPRESENTS THE TRACK NU MBER AND THE NEXT IS THE SECTOR NUMBER. IN OTHER WORDS, IT' S TELLING US THAT THE CATALOG STARTS ON TRACK $11, SECTOR $0F (ON A DOS 3.2 DISK, IT STARTS AT SECTOR $0C). THE FOURTH B YTE, "03" TELLS US THAT THE DOS VERSION IS 3.3 (A "2" = 3.2). THE FIFTH AND SIXTH BYTES AREN'T USED. THE "FE" GIVES US T HE DISK VOLUME NUMBER WHICH IN THIS CASE IS 254 DECIMAL.

THE NEXT SEVERAL ROWS CONTAIN ZEROS BUT WHEN WE GET TO BYTE "$27", NOTICE THE "7A". THIS BYTE INDI CATES THE MAXIMUM NUMBER OF TRACK/SECTOR ADDRESSES THAT A TRACK AND SECTOR LIST IS ALLOWED TO HOLD, SO $7A = 122 (DECIMAL) OF 256 BYTE SECTORS IN A TRACK/SECTOR LIST.

STARTING WITH BYTE $30, YOU WILL SEE TH E FOLLOWING:

```
  BYTE $30....12 01 00 00 23 10 00 01
```

THE FIRST NUMBER, "12", INDICATES THAT TRACK $12 WAS THE LAST TRACK ALLOCATED BY DOS FOR FILE STORAGE . THE NEXT NUMBER MAY BE DIFFERENT, SINCE A SMALL VARIATION I N THE WAY FID WROTE OUT THE FILES COULD ALTER THE SELECTION OF THE NEXT TRACK USED. IF YOU HAVE "01", THAT TELLS US THAT TH E NEXT TRACK THAT DOS WILL ATTEMPT TO WRITE TO IS $12 + $01 = $13. INSTEAD OF A "1", YOU MIGHT SEE AN "FF". THIS TELLS DOS TO SEARCH IN A NEGATIVE DIRECTION, FOR EXAMPLE 12 - 1, FOR

THE NEXT AVAILABLE TRACK. A HEX NUMBER WHOSE HIGH BIT IS SET OR WHOSE VALUE IS $80 OR GREATER IS TAKEN AS A NEGATIVE N UMBER BY THE SYSTEM.

  NOW GO TO THE "23 10". THESE BYTES INDI CATE THAT DOS HAS FOR- MATTED $23(35) TRACKS ON THE DISK AND T HAT EACH TRACK CONTAINS $10(16) SECTORS. THE "00 01" ALSO INDI CATE THAT THERE ARE $100(256) BYTES PER SECTOR. THIS IS WR ITTEN IN LO/HI BYTE FORMAT - SO REVERSE THEM TO GET $100.

BIT MAPS
--- ----

  NOW COMES THE "BIT MAP" FIELD WHICH IS THE MOST IMPORTANT AREA OF THE VTOC. STARTING WITH BYTE $38 AN D EXTENDING UP TO BYTE $C8 ARE THE BIT MAPS. THESE BIT M APS TELL DOS WHICH TRACKS AND SECTORS HAVE NOT BEEN WRITTE N ON AND WHICH ARE AVAILABLE FOR FILE STORAGE. SO DOS DOE SN'T CLOBBER THE PROGRAMS ON THE DISK, IT WILL LOOK AT T HE BIT MAP FIELD EACH TIME YOU SAVE ANY NEW INFORMATION ON TH E DISK. IF YOU'RE LOOKING AT THE "FID"ED SYSTEM MASTER, Y OU WILL BE SEEING MOSTLY ZEROS. SINCE THIS IS A LITTLE D IFFICULT TO LEARN ON TRY THE FOLLOWING:

  (1) BREAK OUT OF YOUR DISK EDIT PROGR AM

  (2) PLACE AN UNINITIALIZED DISK IN TH E DRIVE

  (3) TYPE "CALL -151" TO GO INTO THE M ONITOR

  (4) TYPE "BEFE:24"

  (5) NOW "INIT HELLO" ON THE UNINITIAL IZED DISK

  (6) REBOOT YOUR DISK EDIT PROGRAM AND READ IN TRACK $11, SECTOR $00 OF THE TEST DISK YOU J UST CREATED.

  YOU WILL NOW NOTICE SEVERAL DOZEN "FF"S ' THAT WEREN'T PRESENT ON THE SYSTEM MASTER DISK. NOW LET'S L OOK AT THE ROWS STARTING WITH BYTE $38 WHICH IS THE FIRST BYTE O F THE BIT MAP.

  WHOEVER WROTE DOS, EMPLOYED A SIMPLE WA Y OF RECORDING THE STATUS (FULL OR EMPTY) OF EVERY SINGLE SECTOR ON THE DISK AND SQUEEZING THIS INFO INTO AS LITTLE SPAC E AS POSSIBLE. IT'S DONE BY ASSIGNING A TWO BYTE "MAP" TO E ACH TRACK AND THEN STRUCTURING DOS TO VARY THE VALUES OF T HE MAP WHICH INDICATES THE TRACKS AVAILABLE SECTORS.

  THE ILLUSTRATION BELOW SHOWS HOW THE MA PS ARE LINKED TO THEIR RESPECTIVE TRACKS:

```
BYTE# TRK $00         TRK $01
 38:  !00 00! 00 00 !00 00! 00 00
      TRK $02         TRK $03
 40:  !00 00! 00 00 !FF FF! 00 00
      TRK $04         TRK $05
 48:  !FF FF! 00 00 !FF FF! 00 00
  :
  :
  :   TRK $1E         TRK $1F
 B0:  !FF FF! 00 00 !00 00! 00 00
```

  THE FIRST "00 00" STARTING WITH BYTE $3 8 IS THE BIT MAP FOR TRACK $00, THE

NEXT "00 00" ARE SKIPPED AND THE THIRD BYTE PAIR ARE ASSIGNED TO $01, THIS METHOD C ONTINUES ON THROUGH THE BIT MAP FIELD.

THIS APPEARS AS SEEMINGLY MEANINGLESS D ATA ABOUT EACH SECTOR'S STATUS BUT REMEMBER THAT ALL HEX BYTES ARE COMPOSED OF 8 INDIVIDUAL BITS.   IF YOU ARE UNFAMILIAR WITH THIS, THEN I SUGGEST YOUR READ UP ON THE HEXADECIMAL SYSTEM. I WON'T GO INTO IT HERE BECAUSE THERE ARE NUMEROUS ARTICLES AND BOOKS, ETC., ON THIS SUBJECT. ANYWAY, IF WE A LLOT 2 BYTES FOR EACH TRACKS BIT MAP, WE THEN HAVE 16 BITS TO PLAY WITH.

ISN'T IT A COINCIDENCE THAT EACH TRACK ALSO CONTAINS 16 SEC- TORS. AND WHAT'S EVEN MORE CONVENIENT IS THAT EACH BIT CAN HAVE 1 OR 2 VALUES - A '0' OR A '1'. I F A SINGLE BIT IS ASSIGNED TO EACH SECTOR IN A TRACK, WE CAN SHOW THAT A SECTOR IS FREE BY SETTING ITS BIT TO A '1', WH EREAS A '0' WOULD TELL US AND DOS THAT IT CONTAINS DATA. LOOK AT THE EXAMPLE BELOW:

```
SECTOR #'S : FDECBA98   76543210
BIT VALUES : 00111111   11111111
BYTE VALUES:      3F       FF
```

THE '0'S' IN SECTORS $0F AND $0E TELLS US THAT THESE ARE USED; THE '1'S' IN SECTORS $0D-$00 TELLS US T HAT THEY ARE FREE. THE APPLE WILL AUTOMATICALLY TRANSLATE THE BINARY NUMBER, '00111111' INTO THE HEX NUMBER '3F', AN D '11111111' INTO 'FF', AND THE BIT MAP WOULD SHOW UP AS '3F FF '. REFER TO THE HEX, BINARY, DECIMAL TABLE IN THE FIRST TUTO RIAL.

IF YOU'VE BEEN FOLLOWING ALONG IN THE E XAMPLE ON THE TEST DISK YOU SHOULD SEE THESE 2 BYTES STARTING A T BYTE $80. THIS TELLS US THAT THE '3F FF' REPRESENTS THE STAT US OF THE SECTORS IN TRACK $12(18). THIS IS WHERE THE HELLO PROGRAM WAS LOCATED WHEN YOU INITIALIZED THIS TEST DISK. S O IT OCCUPIES SECTORS $0E AND $0F. WHENEVER DOS WRITES DATA ON A TRACK, IT ALWAYS STARTS WITH THE HIGHEST AVAILABLE SECTO R AND THEN WORKS DOWN.

A GLANCE AT THE TABLE IN THE FIRST TUTO RIAL SHOWS US THAT IF YOU WANT TO SHOW THAT ALL THE 16 SECTOR S ON A GIVEN TRACK ARE FREE, THEN ALL OF ITS 16 BITS ARE SET T O '1', GIVING US AN 'FF FF' BYTE-PAIR. IF YOU WANTED TO RE SERVE AN ENTIRE TRACK FOR SOMETHING, THEN PLACE '00 00' INTO ITS BIT MAP.

NOW THAT WE'RE BECOMING FAMILIAR WITH D OS AND ITS VTOC, NEXT TIME WE'LL START PERFORMING A FEW TIPS & TRICKS.

```
==============================================================================
DOCUMENT crackin.app
==============================================================================

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>                                                       <<
>>                 AN INTRODUCTION TO CRACKING                 <<
>>                                                       <<
>>          A treatise for the neophytes in the Apple world,                 <<
>>          who are full of questions with no one to answer           <<
>>          them.                                        <<
>>                                                       <<
>>                      by The Necromancer                     <<
>>                                                       <<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Greetings to all, and welcome to the wonderful world of Apple software unprotection! Those of you who are long-established hackers and crackers, you will probably not find a lot herein to spark your interest (although you can never be sure) -- this is especially for those new to the field.

This does not necessarily mean that you don't know how to program -- there are many people who are experienced programmers, but have never really gotten into the unprotection racket. In fact, knowing how to program is necessary if you're going to get very far in cracking software. There is no help for it, since the whole point is undoing something that someone else did!

There are a few 'tools of the trade' which, although not absolutely necessary, will make your life a whole lot easier if you have them around. These tools include as one of the most helpful items a monitor ROM. Like I said, it is not *absolutely* necessary, for basic cracking. But if you're going to get a whole lot done, it will become needed. In order to have a monitor ROM, however, you do *not* have to own an Apple II. Those of you with an Apple II+ or Apple //e can come by one in several ways.

The first of these is putting a monitor ROM image in a language card, and write-protecting the language card. This is a somewhat involved hardware modification that I will not go into here, but instructions for it can be found in numerous places.

Another way to get a monitor ROM is by simply buying one and installing it on your motherboard in place of your old Autostart ROM. Or, a similar modification, you can put it on your language card. Some of you may not know that your language card contains an Autostart ROM image, which actually takes precedence over the ROM on the motherboard. However, you can easily construct a switch, which will allow you to choose between your motherboard ROM and your language card ROM. Thus, you can put a monitor ROM on your language card and an Autostart ROM on your motherboard, and switch between them as you like. This is very handy indeed, since you can have either one you want, whenever you want, simply by flicking a switch. The uses of a monitor ROM will be discussed later on.

In any case, the other handy items are all software.   The most important of these is a disk Zap program, some utility for editing a disk sector-by-sector. The best one of these that I have come across is Zap, from Bag of Tricks.

The other useful utilities are a variety of cracking utilities, from the various Muffin-type programs to other disk-viewing programs. The Muffins are for copying programs from protected disks to normal disks, and the disk viewers are for deciphering what on Earth these people have done to their disks.

Now then, down to business. What good is a monitor ROM, some of you may be asking? Well, you should know that when you press reset on an Apple with an Autostart ROM, you are at the mercy of a few memory locations in page 3 of memory. These locations are $3F2-$3F4 (we are going to stick with hexadecimal numbers here -- get used to them, you'll be seein3fqof them!). $3F2 and $3F3 contain the address (lo-byte, hi-byte) to jump to when reset is pressed, and $3F4 contains the exclusive-or of the value in $3F3 with an $A5. This third byte is used by the Apple for checking whether it has just been turned on. If this byte does not contain the XOR of $3F3 with an $A5, when you press reset the monitor will perform a cold start. This is how you can make the machine reboot on a reset, by the way -- simply poke a value like zero into either $3F3 or $3F4.

Anyway, what does all this have to do with a monitor ROM? Well, this dependency of the Autostart ROM makes it easy on software protectors. All they have to do is tell the Apple where to go when the reset key is pressed. With a monitor ROM, you will always go to the same place -- the monitor -- when the reset key is pressed. This means that you are free to go on in and wade about in their code, to decipher what they're doing.

By the way, for reference's sake, there is another location which is handy to know about, which is the Applesoft run flag at $D6. If this is set, any command given to the DOS parser will cause the program in memory to be run. This is a common location to set, so if you are attempting to crack a basic program, it is likely to be set. To defeat it, simply set it to any value less than 128.

With a monitor ROM, some programs become a cinch to crack. Basically, any single-loading program (usually games) can almost always be cracked simply by pressing reset and rebooting onto another disk. Some notes, however...

Before you can do anything with it, you have to know how it runs. Say you've got Program X, and you've pressed reset into the monitor. It is a single-loading game, so all of it is in memory there somewhere. Where does it start? Good question.

A frequent place is at $800, or sometimes $7FD, three bytes before $800. Try an 800G in the monitor. If it starts up, great! If not, time to look again. Try the various page boundaries, particularly $2000, $4000, $6000, etc. Check the hires pages with a C050 <c/r>, C057 <c/r> to see the first hi-res page. If it has a title picture, the program isn't there. Try C055 <c/r> to see page 2 of hi-res. If there is a picture on page one and not on page two, $4000 is a very possible starting location.

There are hints for finding the starting location of a program. Look for a sequence that will turn on the hi-res pages for display -- look for addresses like $C050, $C055, $C052, the graphics soft switches. Look for a keyboard read -- games will often show a title picture and wait for a keypress, reading the strobe at $C000. If none of these turns up an9thing, then it may be necessary to try some likely places at random -- it can turn up useful information sometimes, although it's not exactly recommended practice. Look for initialization routines, or jump tables.

If all of this fails, then perhaps the protectors have tried some sneakier tricks, which will be gone into in later columns.

Once you have found the starting location, then what? Then it's time to transfer the program to your own disk. Remember one of the prime rules of cracking -- when working, always have one or two blank, initialized disks handy, with a normal slave DOS on them.

Let's say you have found the starting location to Program X -- what to do? Well, let's look at memory for a moment. Free memory starts, basically, at $800, above the text page (it is possible to use this area, but that's a subject for later), and goes until $9600, on a normal disk.

However, it is more than likely that this disk you're cracking has no DOS. That upper limit of $9600 is for a disk with normal DOS. Assuming this program is a single-loading game, it undoubtedly has no DOS. Thus, this program is free to go until $BFFF, really.

But if you boot your slave disk now, it will wipe out memory from $800 to $900, and $9600-$BFFF. Therefore, we must split Program X into smaller pieces. The first piece is from $800-$4000. To put it onto your disk, first move it up to protect it from your booting. Move it up to $4000 with a *4000<800.3FFFM. This moves everything from 800 to 3FFF to 4000 up. Then do a 6<control-P> to reboot.

Now save segment one of the program to disk, after moving it down:

```
CALL -151
800<4000.7800M
BSAVE PROGRAM X (800-3FFF),A$800,L$3800
```

And you have a good part of the program. Now reboot the Program X disk, and press reset again. Now to save the rest. We are going to assume that Program X only goes up to $9600, to make life easy for now. So just reboot4again, and save part two of Program X with a BSAVE PROGRAM X (4000-9600),A$4000,L$5600.

At this point, test your Program X by BLOADing the two pieces and running it. If it still works, you're in business. If not, the likelihood is that the program requires some other pieces of memory. Either the piece it needs is below $800, or above $9600, obviously. Try checking the code near the entry point, and see if you can find any clues to what locations it might access. In either case, though, it becomes more complicated, since you can't just BRUN something that requires memory below $400 or above $9600. If it does not use memory much above $9600, note that you can save over 1K with a maxfiles command, since from $9600 to $9D00 are the DOS buffers.

Assuming the program works, you just have the chore of cutting down the size by figuring out what is really necessary of what you just saved. Once you have done that, you can just save the whole thing into a single file, give yourself credit, and give the program to everyone you know.

One more item: if the program becomes greater than $7FFF in length, DOS will not let you save it in one file. Change location A964 in DOS to $FF and you won't have any problems (why this restriction is there, *I* don't know!).

Next time I will get into DOS and what modifications there are to help in cracking........

May your cracks be forever successful!

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>The Necromancer<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>                                                          <<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>The Cracker's Guild<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
==============================================================================
DOCUMENT crakowit.app
==============================================================================

               ***************************************
               *                            *
               *                            *
               *  KRAKOWICZ'S KRACKING KORNER IV   *
               *                            *
               *                            *
               *                            *
               *                            *
               *                            *
               *        THE ARCADE MACHINE       *
               *                            *
               *                            *
               *                            *
               * WITH NOTES ON NMI AND IDSI'S JUGGLER*
               *                            *
               *                            *
               *                            *
               ***************************************
```

AFTER A NINE-MONTH DELAY, BR0DERBUND HAS FINALLY RELEASED THE ARCADE MACHINE (A.M.). THE PROTECTION SCHEME IS A NEW CHALLENGE FOR COPIERS, SINCE IT USES THE TECHNIQUE KNOWN AS SPIRALING OR QUARTER-TRACKING, AS WELL AS THE STANDARD BR0DERBUND SYSTEM OF A NEW ADDRESS MARKER FOR EACH TRACK. AN ATTEMPT TO COPY THE DISK WITH A CONVENTIONAL NIBBLE COPIER QUICKLY REVEALS THAT TRACKS 0 AND 3-11 ARE EASILY COPIED WITH AN ADDRESS MARKER OF D5 AA 96, WHILE THE REST OF THE TRACKS ARE A MYSTERY. PROBING INTO THE LOADER REVEALS THE FOLLOWING INFORMATION ABOUT TRACK USAGE:

```
      TRACK CONTENTS
      ----- --------

      T0/S0 PRELOADER --> 800-8FF
            (AS ALWAYS)
      /S1-5 LOADER --> 300-7FF

      T1-2  HIRES SPLIT "BR0DERBUND"
             LOGO AND PROGRAM

      T12-20      MAIN PROGRAM WHICH LOADS
             INTO 800-BFFF

      T12-13.5    FOUR HALFTRACKS USED FOR
             QUARTER-TRACKING

      T3-4  #1 SHAPE CREATOR

      T5-6  #2 PATH CREATOR

      T7-8  #3 GAME OPTIONS

      T9-A  #4 LEVEL OPTIONS

      TC-D  #5 BKGD/TITLE CREATOR
```

```
     TE-F  #6 LOAD/SAVE GAME

     T10-11      #7 CREATE GAME DISK

           (OPTION #8 JUMPS TO 0800
            TO RUN THE GAME)
```

THE APPROACH TO KRACKING THIS TYPE OF PROGRAM SEEMS STRAIGHTFORWARD:LOAD THE PROGRAM INTO MEMORY, RESET IT, AND SAVE IT OUT TO DISK AS A BINARY FILE, WITH THE APPROPRIATE MEMORY MOVES. HOPEFULLY, YOU'LL LOCATE THE STARTING ADDRESS AND BE ABLE TO RUN THE BINARY FILE AT WILL. IF YOU WISH TO INCLUDE ALL OF THE ADVERTISING FOR BR0DERBUND AT THE BEGINNING, THIS WORKS. IF YOU TRY TO DELETE THE DUAL BANNER, IT CRASHES. THE REASON IS THAT MODULE SWITCHING IS VIA THE STACK--THEY PUSH THE CORRECT LOCATION ONTO THE STACK AND DO AN RTS. SO, UNLESS YOU HAPPEN TO KNOW THE VALUE OF THE PROGRAM COUNTER (THAT IS, EXACTLY WHAT THE ADDRESS WAS WHEN YOU STOPPED), THE STACK POINTER (S) AND THE PROCESSOR STATUS WORD (P), AND RESTORE THEM EXACTLY AS THEY WERE BEFORE THE RESET, THE PROGRAM PROBABLY WON'T RUN. ANYONE WHO TRIED TO BREAK JUGGLER FOUND THIS TO BE FRUSTRATING IN THE EXTREME, SINCE SOMETIMES THE GAME WOULD RUN ALL THE WAY THROUGH THE FIRST LEVEL BEFORE CRASHING - THE SAME TECHNIQUE WAS USED THERE, BUT WITH EVEN MORE PROTECTION.

THERE IS A HARD WAY AND AN EASY WAY TO DO EVERYTHING, AND IF YOU ARE COMPLETELY RESTRICTED TO SOFTWARE DEVICES, IT IS STILL POSSIBLE TO BREAK ARCADE MACHINE. REFERRING TO THE NIBBLE ALTERATION TECHNIQUES DESCRIBED IN THE PREVIOUS EPISODE, IT IS POSSIBLE TO LOCATE AND ALTER THE GAME LOADER SO THAT IT HALTS WITH CONDITIONS WELL DEFINED AFTER THE ENTIRE PROGRAM IS IN MEMORY. IF IT IS YOUR PURPOSE IN LIFE TO LEARN AS MUCH AS YOU POSSIBLY CAN ABOUT DISK PROTECTION SCHEMES AND THE CIRCUMVENTION THEREOF (ONLY A FEW REALLY CRAZY PEOPLE ARE SO INCLINED), THIS IS REWARDING. IF YOU ARE INTERESTED IN PREPARING AN UNPROTECTED VERSION OF THE GAME WITH MINUMUM ADVERTISING AND MINIMUM EFFORT, HOWEVER, THERE IS AN EASIER WAY.

THIS SOLUTION IS ELEGANT, BUT REQUIRES A VISIT TO THAT GOD OF THE UNDERWORLD =>HARDWARE<=. B

```
 PLEASE PLACE ANY NEW KRAKING TIPS
 OR TECNIQUES ON THE KRACKING BOARD.

  [\/][\/][\/][\/][\/][\/][\/][\/][\/]

          USS ENTERPRISE I
            318-367-8860
```

```
===============================================================================
DOCUMENT cramit.app
===============================================================================

==============================
PROGRAM COMPRESSION
"HOW TO" AND A UTILITY PROGRAM
==============================

  WITH IN MANY CRACKED PROGRAMS LURK MANY WAISTFUL REPETITIONS OF DATA (I.E.  23
04 59 55 55 55 55 55 55 55 55) THEY ARE NECCESSARY FOR THE PROGRAM TO WORK BUT
THEY TAKE UP A LOT OF SPACE.  TO SOLVE THIS "PROBLEM" I HAVE WRITTEN A PROGRAM
TO CRAM TOGETHER OTHER PROGRAMS AND A PROGRAM TO UNCRAM AND RUN THE ORIGINAL
PROGRAM.  OF COURSE THERE ARE MANY VERY COMPLICATED WAYS TO "GET THE MOST OUT OF
EVERY BYTE" BUT THE WAY I CHOSE IS VERY SIMPLE:

  IF THERE IS A REPETITION IT IS STORED LIKE THIS:

  1) NUMBER OF REPEATS (UP TO $FF), REPEAT VALUE

  OTHERWISE STORE THINGS LIKE THIS:

  2) $00, NUMBER OF UNIQUES (UP TO $FFFF), LIST OF UNIQES

  USING THIS METHOD THE ABOVE EXAMPLE WOULD LOOK LIKE THIS:

  00 00 03 23 04 59 07 55

  IN THIS SMALL EXAMPLE NOT MUCH WAS SAVED BUT IN REAL LIFE HOWEVER I HAVE BEEN
ABLE TO REDUCE THE SIZE BY 4K TO 12K DEPENDING ON THE PROGRAM USING THIS SIMPLE
METHOD!

  THE FOLLOWING TWO PROGRAMS ARE RELOCATABLE SO THEY CAN BE RUN ANYWHERE YOU
WISH.  THE FIRST OF THE TWO IS THE CRAMMER AND THE SECOND PROGRAM IS THE
UNCRAMMER.

  TO USE THE CRAMMER JUST:

  1) LOAD THE PROGRAM TO BE CRAMMED ALONG WITH THE CRAMMER (AT LEAST ONE PAGE
ABOVE THE PROGRAM)

  2) ENTER THE FOLLOWING INTO PAGE 0:

     00- PROGRAM START (LSB)
     01- PROGRAM START (MSB)
     02- PROGRAM END (LSB)
     03- PROGRAM END (MSB)

  3) RUN CRAMMER
  4) GET THE FOLLOWING FROM PAGE 0:

     00- CRAMMED START (LSB)
     01- CRAMMED START (MSB)
     02- CRAMMED END (LSB)
     03- CRAMMED END (MSB)

BEFORE==>     PPPPPPPPPPPPP      CRAM
```

```
            !       !      !  !
DURING==> CCCCCCC   !      CRAM
        !   ! !     !      !  !
AFTER===> !   ! !    CCCCCCC     CRAM
        !   ! !    !      !!     !  !
      $0400   ! !    !    !!     !  !
(PROGRAM START) !    !    !!     !  !
          $????    !      !!     !  !
        (CRAM START)      !!     !  !
            (PROGRAM END)!       !  !
              (CRAM END)  (CRAMMER)
```

TO USE THE UNCRAMMER JUST:

  1) LOAD CRAMMED PROGRAM AND UNCRAMMER RIGHT ABOVE IT
  2) CHANGE THE NOP'S IN UNCRAM TO THE FOLLOWING:

```
     NOP- CRAMMED START (LSB)
     NOP- CRAMMED START (MSB)
     NOP- CRAMMED END (LSB)
     NOP- CRAMMED END (MSB)
     NOP- ORIGINAL PROGRAM START (LSB)
     NOP- ORIGINAL PROGRAM START (MSB)
```

  3) CHANGE THE JMP $FF69 IN UNCRAM TO THE STARTING ADDRESS OF THE ORIGINAL PROGRAM.

  4) PUT A JMP $(UNCRAM ADDRESS) BEFORE CRAMMED START AND YOUR DONE

```
BEFORE===>    CCCCCCCC:UNCRAM
              !    !
AFTER====>  PPPPPPPPPPPPP!
          !   !   !!
(PROGRAM START)   !     !!
       (CRAM START)     !!
          (PROGRAM END)!
            (CRAM END)
```

AN EXAMPLE:

 CANNONBAL BLITZ-

  MY BLITZ LOADED AT $0800 TO $8FFF AND THE STARTING ADDRESS WAS $2900 I THEN LOADED CRAMMER AT $B000 AND TYPED:

```
  00:00 08 FF 8F

  B000G

  0 <RETURN> <RETURN>

     0000- 77 15 FF 90 (A SAVING OF 3K)
```

  THE CRAMMED PROGRAM NOW RESIDES AT $1577 TO $90FF. I THEN LOADED UNCRAM AT $9100 AND TYPED:

```
  911D:77 15 FF 90 00 08
  913B:4C 00 29
  1574:4C 00 91

  THE NEW BLITZ STARTS AT $1574 AND ENDS AT $919C
```

THE CRAMMER:

```
1000:A0 00 A9 00 85 04 A9 04
1008:85 05 B1 02 C8 AA E8 8A
1010:91 02 88 A5 00 85 06 A5
1018:01 85 07 B1 00 A2 00 85
1020:0C A5 06 C5 02 D0 18 A5
1028:07 C5 03 D0 12 8A F0 43
1030:91 04 E6 04 D0 02 E6 05
1038:A5 0C 91 04 18 90 7B A5
1040:0C E6 06 D0 02 E6 07 D1
1048:06 D0 04 E8 D0 D1 CA 85
1050:0C 8A C9 06 90 1D 91 04
1058:E6 04 D0 02 E6 05 A5 0C
1060:91 04 E6 04 D0 02 E6 05
1068:A5 06 85 00 A5 07 85 01
1070:18 90 A0 A9 00 85 0A 85
1078:0B 91 04 E6 04 D0 02 E6
1080:05 A5 04 85 08 A5 05 85
1088:09 E6 04 D0 02 E6 05 E6
1090:04 D0 02 E6 05 B1 00 91
1098:04 E6 0A D0 02 E6 0B A5
10A0:00 C5 02 D0 19 A5 01 C5
10A8:03 D0 13 A5 0B 91 08 E6
10B0:08 D0 02 E6 09 A5 0A 91
10B8:08 18 90 40 90 B3 E6 00
10C0:D0 02 E6 01 E6 04 D0 02
10C8:E6 05 A5 00 85 06 A5 01
10D0:85 07 B1 00 A2 00 E6 06
10D8:D0 02 E6 07 D1 06 D0 04
10E0:E8 D0 F3 CA 85 0C 8A C9
10E8:06 90 AA A5 0B 91 08 E6
10F0:08 D0 02 E6 09 A5 0A 91
10F8:08 18 90 C0 E6 03 A5 02
1100:85 06 A5 03 85 07 B1 04
1108:91 06 C6 06 A5 06 C9 FF
1110:D0 02 C6 07 C6 04 A5 04
1118:C9 FF D0 02 C6 05 A5 04
1120:C9 FF D0 E2 A5 05 C9 03
1128:D0 DC E6 06 D0 02 E6 07
1130:A5 06 85 00 A5 07 85 01
1138:60
```

THE UNCRAMMER:

```
2000:20 58 FF BA BD 00 01 85
2008:07 CA BD 00 01 85 06 A2
2010:05 A0 20 B1 06 95 00 88
2018:CA D0 F8 F0 06 EA EA EA
2020:EA EA EA B1 06 95 00 A0
2028:00 E6 02 D0 02 E6 03 A5
2030:00 C5 02 D0 09 A5 01 C5
2038:03 D0 03 4C 69 FF B1 00
2040:E6 00 D0 02 E6 01 09 00
```

```
2048:D0 3B B1 00 85 07 E6 00
2050:D0 02 E6 01 B1 00 85 06
2058:E6 00 D0 02 E6 01 B1 00
2060:91 04 E6 00 D0 02 E6 01
2068:E6 04 D0 02 E6 05 C6 06
2070:A5 06 C9 FF D0 02 C6 07
2078:A9 00 C5 07 D0 E0 C5 06
2080:D0 DC 18 90 AA AA B1 00
2088:E6 00 D0 02 E6 01 E8 91
2090:04 E6 04 D0 02 E6 05 CA
2098:D0 F5 18 90 92
```

JOHN
RAYMONDS

```
============================================================================
DOCUMENT cramit.txt
============================================================================

3

---------------------------------------
[Ctrl-S pauses/Space=quit]

3
0

   THERE ARE TWO KNOWN WAYS TO WRITE
PROTECT A RAMCARD. THIS ONE INVOLVES
USING A SWITCH RIGHT ON THE RAMCARD. TO
DO THIS YOU MUST HAVE A LITTLE KNOW HOW
OF THE RAM CARD OR OWN A SCHEMATIC OF
YOUR RAM CARD. YOU MUST BE ABLE TO FIND
THE R/W LINE. FIRST

---------------------------------------

Enter (1-10, M=Menu, Q=Quit) :4

---------------------------------------
[Ctrl-S pauses/Space=quit]

4
0

     HOW TO MODIFY PRO-DOS TO WORK
** WITH ANY ROM & OTHER PRO-DOS INFO **

  FIRST, MAKE A COPY OF THE DISK USING
COPYA FROM THE DOS 3.3 MASTER. THEN
USING ANY 16 SECTOR DISK-ZAP UTILITY
THE FOLLOWING SECTORS ARE CHANGED:

        TRACK 1, SECTOR A: BYTES E8
THROUGH E9 BECOME "EA"
        TRACK 1, SECTOR C: BYTES 5F
THROUGH 71 BECOME "EA"

        THE ROUTINES AT THESE LOCATIONS
WERE LOOKING FOR "SIGNATURE" BYTES
IN THE F8 MONIT

---------------------------------------

Enter (1-10, M=Menu, Q=Quit) :

Enter (1-10, M=Menu, Q=Quit) :5

---------------------------------------
[Ctrl-S pauses/Space=quit]

5
```

```
0

****************************************
**                                    **
**       A NEW 'DOS' TIP TO ADD       **
**         DISK SPACE TO YOUR         **
**           DOS 3.3  DISKS           **
**                                    **
**        BY: MARC EPSTEIN            **
**                                    **
*****************

---------------------------------------

Enter (1-10, M=Menu, Q=Quit) :M8

---------------------------------------
[Ctrl-S pauses/Space=quit]
```

PROGRAM COMPRESSION
"HOW TO" AND A UTILITY PROGRAM
==============================

WITH IN MANY CRACKED PROGRAMS LURK MANY WAISTFUL REPETITIONS OF DATA (I.E. 23 04 59 55 55 55 55 55 55 55 55) THEY ARE NECCESSARY FOR THE PROGRAM TO WORK BUT THEY TAKE UP A LOT OF SPACE. TO SOLVE THIS "PROBLEM" I HAVE WRITTEN A PROGRAM TO CRAM TOGETHER OTHER PROGRAMS AND A PROGRAM TO UNCRAM AND RUN THE ORIGINAL PROGRAM. OF COURSE THERE ARE MANY VERY COMPLICATED WAYS TO "GET THE MOST OUT OF EVERY BYTE" BUT THE WAY I CHOSE IS VERY SIMPLE:

IF THERE IS A REPETITION IT IS STORED LIKE THIS:

1) NUMBER OF REPEATS (UP TO $FF),
   REPEAT VALUE

OTHERWISE STORE THINGS LIKE THIS:

2) $00, NUMBER OF UNIQUES (UP TO
   $FFFF), LIST OF UNIQES

USING THIS METHOD THE ABOVE EXAMPLE WOULD LOOK LIKE THIS:

```
00 00 03 23 04 59 07 55
```

IN THIS SMALL EXAMPLE NOT MUCH WAS SAVED BUT IN REAL LIFE HOWEVER I HAVE BEEN ABLE TO REDUCE THE SIZE BY 4K TO 12K DEPENDING ON THE PROGRAM USING THIS SIMPLE METHOD!

THE FOLLOWING TWO PROGRAMS ARE
RELOCATABLE SO THEY CAN BE RUN ANYWHERE
YOU WISH. THE FIRST OF THE TWO IS THE
CRAMMER AND THE SECOND PROGRAM IS THE
UNCRAMMER.

TO USE THE CRAMMER JUST:

1) LOAD THE PROGRAM TO BE CRAMMED
   ALONG WITH THE CRAMMER (AT LEAST
   ONE PAGE ABOVE THE PROGRAM)

2) ENTER THE FOLLOWING INTO PAGE 0:

   00- PROGRAM START (LSB)
   01- PROGRAM START (MSB)
   02- PROGRAM END (LSB)
   03- PROGRAM END (MSB)

3) RUN CRAMMER

4) GET THE FOLLOWING FROM PAGE 0:

   00- CRAMMED START (LSB)
   01- CRAMMED START (MSB)
   02- CRAMMED END (LSB)
   03- CRAMMED END (MSB)

```
BEFORE==>     PPPPPPPPPPPPP        CRAM
              !           !        !  !
DURING==> CCCCCCC         !        CRAM
          !   ! !         !        !  !
AFTER===> !   ! !    CCCCCCC       CRAM
          !   ! !    !    !!       !  !
      $0400   ! !    !    !!       !  !
(PROGRAM START) !    !    !!       !  !
            $????    !    !!       !  !
          (CRAM START)    !!       !  !
              (PROGRAM END)!       !  !
                  (CRAM END)  (CRAMMER)
```

TO USE THE UNCRAMMER JUST:

1) LOAD CRAMMED PROGRAM AND UNCRAMMER
   RIGHT ABOVE IT

2) CHANGE THE NOP'S IN UNCRAM TO THE
   FOLLOWING:

   NOP- CRAMMED START (LSB)
   NOP- CRAMMED START (MSB)
   NOP- CRAMMED END (LSB)
   NOP- CRAMMED END (MSB)
   NOP- ORIGINAL PROGRAM START (LSB)
   NOP- ORIGINAL PROGRAM START (MSB)

3) CHANGE THE JMP $FF69 IN UNCRAM TO

```
      THE STARTING ADDRESS OF THE
      ORIGINAL PROGRAM.

   4) PUT A JMP $(UNCRAM ADDRESS) BEFORE
      CRAMMED START AND YOUR DONE

BEFORE===>         CCCCCCCC:UNCRAM
                   !      !
AFTER====>  PPPPPPPPPPPPP!
            !     !     !!
(PROGRAM START)   !     !!
       (CRAM START)     !!
            (PROGRAM END)!
                (CRAM END)

AN EXAMPLE:

 CANNONBAL BLITZ-

MY BLITZ LOADED AT $0800 TO $8FFF
AND THE STARTING ADDRESS WAS $2900
I THEN LOADED CRAMMER AT $B000 AND
TYPED:

  00:00 08 FF 8F

  B000G

  0 <RETURN> <RETURN>

     0000- 77 15 FF 90 (A SAVING OF 3K)

THE CRAMMED PROGRAM NOW RESIDES AT
$1577 TO $90FF.  I THEN LOADED UNCRAM
AT $9100 AND TYPED:

  911D:77 15 FF 90 00 08
  913B:4C 00 29
  1574:4C 00 91

THE NEW BLITZ STARTS AT $1574 AND ENDS
AT $919C

THE CRAMMER:

1000:A0 00 A9 00 85 04 A9 04
1008:85 0
5 B1 02 C8 AA E8 8A
1010:91 02 88 A5 00 85 06 A5
1018:01 85 07 B1 00 A2 00 85
1020:0C A5 06 C5 02 D0 18 A5
1028:07 C5 03 D0 12 8A F0 43
1030:91 04 E6 04 D0 02 E6 05
1038:A5 0C 91 04 18 90 7B A5
1040:0C E6 06 D0 02 E6 07 D1
1048:06 D0 04 E8 D0 D1 CA 85
1050:0C 8A C9 06 90 1D 91 04
1058:E6 04 D0 02 E6 05 A5 0C
```

```
1060:91 04 E6 04 D0 02 E6 05
1068:A5 06 85 00 A5 07 85 01
1070:18 90 A0 A9 00 85 0A 85
1078:0B 91 04 E6 04 D0 02 E6
1080:05 A5 04 85 08 A5 05 85
1088:09 E6 04 D0 02 E6 05 E6
1090:04 D0 02 E6 05 B1 00 91
1098:04 E6 0A D0 02 E6 0B A5
10A0:00 C5 02 D0 19 A5 01 C5
10A8:03 D0 13 A5 0B 91 08 E6
10B0:08 D0 02 E6 09 A5 0A 91
10B8:08 18 90 40 90 B3 E6 00
10C0:D0 02 E6 01 E6 04 D0 02
10C8:E6 05 A5 00 85 06 A5 01
10D0:85 07 B1 00 A2 00 E6 06
10D8:D0 02 E6 07 D1 06 D0 04
10E0:E8 D0 F3 CA 85 0C 8A C9
10E8:06 90 AA A5 0B 91 08 E6
10F0:08 D0 02 E6 09 A5 0A 91
10F8:08 18 90 C0 E6 03 A5 02
1100:85 06 A5 03 85 07 B1 04
1108:91 06 C6 06 A5 06 C9 FF
1110:D0 02 C6 07 C6 04 A5 04
1118:C9 FF D0 02 C6 05 A5 04
1120:C9 FF D0 E2 A5 05 C9 03
1128:D0 DC E6 06 D0 02 E6 07
1130:A5 06 85 00 A5 07 85 01
1138:60
```

THE UNCRAMMER:

```
2000:20 58 FF BA BD 00 01 85
2008:07 CA BD 00 01 85 06 A2
2010:05 A0 20 B1 06 95 00 88
2018:CA D0 F8 F0 06 EA EA EA
2020:EA EA EA B1 06 95 00 A0
2028:00 E6 02 D0 02 E6 03 A5
2030:00 C5 02 D0 09 A5 01 C5
2038:03 D0 03 4C 69 FF B1 00
2040:E6 00 D0 02 E6 01 09 00
2048:D0 3B B1 00 85 07 E6 00
2050:D0 02 E6 01 B1 00 85 06
2058:E6 00 D0 02 E6 01 B1 00
2060:91 04 E6 00 D0 02 E6 01
2068:E6 04 D0 02 E6 05 C6 06
2070:A5 06 C9 FF D0 02 C6 07
2078:A9 00 C5 07 D0 E0 C5 06
2080:D0 DC 18 90 AA AA B1 00
2088:E6 00 D0 02 E6 01 E8 91
2090:04 E6 04 D0 02 E6 05 CA
2098:D0 F5 18 90 92
```

JOHN
RAYMONDS

```
---------------------------------------
Enter (1-10, M=Menu, Q=Quit) :
```

```
==============================================================================
DOCUMENT crammin.app
==============================================================================

               THE SOUTH POLE..........[312] 677-7140
               *=----------------------------------=*
               *       PROGRAM COMPRESSION       *
               *=----------------------------------=*
```

WITH IN MANY CRACKED PROGRAMS LURK MANY WAISTFUL REPETITIONS OF DATA (I.E. 23 04 59 55 55 55 55 55 55 55 55) THEY ARE NECCESSARY FOR THE PROGRAM TO WORK BUT THEY TAKE UP A LOT OF SPACE. TO SOLVE THIS "PROBLEM" I HAVE WRITTEN A PROGRAM TO CRAM TOGETHER OTHER PROGRAMS AND A PROGRAM TO UNCRAM AND RUN THE ORIGINAL PROGRAM. OF COURSE THERE ARE MANY VERY COMPLICATED WAYS TO "GET THE MOST OUT OF EVERY BYTE" BUT THE WAY I CHOSE IS VERY SIMPLE:

IF THERE IS A REPETITION IT IS STORED LIKE THIS:

1) NUMBER OF REPEATS (UP TO $FF), REPEAT VALUE

OTHERWISE STORE THINGS LIKE THIS:

2) $00, NUMBER OF UNIQUES (UP TO $FFFF), LIST OF UNIQES USING THIS METHOD THE ABOVE EXAMPLE WOULD LOOK LIKE THIS: 00 00 03 23 04 59 07 55

IN THIS SMALL EXAMPLE NOT MUCH WAS SAVED BUT IN REAL LIFE HOWEVER I HAVE BEEN ABLE TO REDUCE THE SIZE BY 4K TO 12K DEPENDING ON THE PROGRAM USING THIS SIMPLE METHOD!

THE FOLLOWING TWO PROGRAMS ARE RELOCATABLE SO THEY CAN BE RUN ANYWHERE YOU WISH. THE FIRST OF THE TWO IS THE CRAMMER AND THE SECOND PROGRAM IS THE UNCRAMMER.

TO USE THE CRAMMER JUST:

1) LOAD THE PROGRAM TO BE CRAMMED ALONG WITH THE CRAMMER (AT LEAST ONE PAGE ABOVE THE PROGRAM)

2) ENTER THE FOLLOWING INTO PAGE 0:

```
      00- PROGRAM START (LSB)
      01- PROGRAM START (MSB)
      02- PROGRAM END (LSB)
      03- PROGRAM END (MSB)
```

3) RUN CRAMMER

4) GET THE FOLLOWING FROM PAGE 0:

```
      00- CRAMMED START (LSB)
      01- CRAMMED START (MSB)
      02- CRAMMED END (LSB)
      03- CRAMMED END (MSB)

BEFORE==>     PPPPPPPPPPPPP      CRAM
            !       !      !  !
DURING==> CCCCCCC   !      CRAM
```

```
        !   ! !     !      !  !
AFTER===> !   ! !    CCCCCCC     CRAM
        !   ! !    !      !!     !  !
      $0400   ! !    !    !!     !  !
(PROGRAM START) !    !    !!     !  !
          $????    !      !!     !  !
        (CRAM START)      !!     !  !
            (PROGRAM END)!       !  !
              (CRAM END)  (CRAMMER)
```

TO USE THE UNCRAMMER JUST:

   1) LOAD CRAMMED PROGRAM AND UNCRAMMER RIGHT ABOVE IT

   2) CHANGE THE NOP'S IN UNCRAM TO THE FOLLOWING:

```
     NOP- CRAMMED START (LSB)
     NOP- CRAMMED START (MSB)
     NOP- CRAMMED END (LSB)
     NOP- CRAMMED END (MSB)
     NOP- ORIGINAL PROGRAM START (LSB)
     NOP- ORIGINAL PROGRAM START (MSB)
```

   3) CHANGE THE JMP $FF69 IN UNCRAM TO THE STARTING ADDRESS OF THE ORIGINAL PROGRAM.

   4) PUT A JMP $(UNCRAM ADDRESS) BEFORE CRAMMED START AND YOUR DONE

```
BEFORE===>    CCCCCCCC:UNCRAM
              !    !
AFTER====>  PPPPPPPPPPPP!
          !   !   !!
(PROGRAM START)   !     !!
       (CRAM START)     !!
          (PROGRAM END)!
            (CRAM END)
```

AN EXAMPLE:

 CANNONBAL BLITZ-

   MY BLITZ LOADED AT $0800 TO $8FFF AND THE STARTING ADDRESS WAS $2900 I THEN LOADED CRAMMER AT $B000 AND TYPED:

```
  00:00 08 FF 8F

  B000G

  0 <RETURN> <RETURN>

     0000- 77 15 FF 90 (A SAVING OF 3K)
```

   THE CRAMMED PROGRAM NOW RESIDES AT $1577 TO $90FF. I THEN LOADED UNCRAM AT $9100 AND TYPED:

```
  911D:77 15 FF 90 00 08
  913B:4C 00 29
  1574:4C 00 91
```

   THE NEW BLITZ STARTS AT $1574 AND ENDS AT $919C.

THE CRAMMER:

```
1000:A0 00 A9 00 85 04 A9 04
1008:85 05 B1 02 C8 AA E8 8A
1010:91 02 88 A5 00 85 06 A5
1018:01 85 07 B1 00 A2 00 85
1020:0C A5 06 C5 02 D0 18 A5
1028:07 C5 03 D0 12 8A F0 43
1030:91 04 E6 04 D0 02 E6 05
1038:A5 0C 91 04 18 90 7B A5
1040:0C E6 06 D0 02 E6 07 D1
1048:06 D0 04 E8 D0 D1 CA 85
1050:0C 8A C9 06 90 1D 91 04
1058:E6 04 D0 02 E6 05 A5 0C
1060:91 04 E6 04 D0 02 E6 05
1068:A5 06 85 00 A5 07 85 01
1070:18 90 A0 A9 00 85 0A 85
1078:0B 91 04 E6 04 D0 02 E6
1080:05 A5 04 85 08 A5 05 85
1088:09 E6 04 D0 02 E6 05 E6
1090:04 D0 02 E6 05 B1 00 91
1098:04 E6 0A D0 02 E6 0B A5
10A0:00 C5 02 D0 19 A5 01 C5
10A8:03 D0 13 A5 0B 91 08 E6
10B0:08 D0 02 E6 09 A5 0A 91
10B8:08 18 90 40 90 B3 E6 00
10C0:D0 02 E6 01 E6 04 D0 02
10C8:E6 05 A5 00 85 06 A5 01
10D0:85 07 B1 00 A2 00 E6 06
10D8:D0 02 E6 07 D1 06 D0 04
10E0:E8 D0 F3 CA 85 0C 8A C9
10E8:06 90 AA A5 0B 91 08 E6
10F0:08 D0 02 E6 09 A5 0A 91
10F8:08 18 90 C0 E6 03 A5 02
1100:85 06 A5 03 85 07 B1 04
1108:91 06 C6 06 A5 06 C9 FF
1110:D0 02 C6 07 C6 04 A5 04
1118:C9 FF D0 02 C6 05 A5 04
1120:C9 FF D0 E2 A5 05 C9 03
1128:D0 DC E6 06 D0 02 E6 07
1130:A5 06 85 00 A5 07 85 01
1138:60
```

THE UNCRAMMER:

```
2000:20 58 FF BA BD 00 01 85
2008:07 CA BD 00 01 85 06 A2
2010:05 A0 20 B1 06 95 00 88
2018:CA D0 F8 F0 06>EA EA EA
2020:EA EA EA<B1 06 95 00 A0
2028:00 E6 02 D0 02 E6 03 A5
2030:00 C5 02 D0 09 A5 01 C5
2038:03 D0 03>4C 69 FF<B1 00
2040:E6 00 D0 02 E6 01 09 00
2048:D0 3B B1 00 85 07 E6 00
```

```
2050:D0 02 E6 01 B1 00 85 06
2058:E6 00 D0 02 E6 01 B1 00
2060:91 04 E6 00 D0 02 E6 01
2068:E6 04 D0 02 E6 05 C6 06
2070:A5 06 C9 FF D0 02 C6 07
2078:A9 00 C5 07 D0 E0 C5 06
2080:D0 DC 18 90 AA AA B1 00
2088:E6 00 D0 02 E6 01 E8 91
2090:04 E6 04 D0 02 E6 05 CA
2098:D0 F5 18 90 92
```

```
==============================================================================
DOCUMENT crisis.app
==============================================================================

MSG LEFT BY: DOCTOR WHO
DATE POSTED: FRI MAR  3  2:52:40 PM

TO CRACK CRISIS MOUNTAIN:

BOOT DOS 3.3
 CALL-151
 B925:18 60
 B988:18 60
 BE48:18
 B942:18
 BAAA:00
 RUN COPYA
COPY CRISIS MOUNTAIN
WITH A SECTOR EDITOR MAKE THE FOLLOWING
CHANGES ON TRACK 0 SECTOR 5
    24:D5 (WAS EB)
    2D:AA (WAS D5)
    36:96 (WAS AA)

NOW ITS CRACKED

ANOTHER CRACK FROM-

------------=> DOCTOR WHO <=-----------

MSG LEFT BY: DOCTOR WHO
DATE POSTED: SAT MAR  4 12:28:38 PM

  TO CRACK DUNGEON & THESEUS AND THE MINO TAUR BY TSR, I HAVE A METHOD THAT
REQUI RES NO WIERD HARDWARE OR EXTRA CARDS.  THIS PROGRAM IS WRITTEN IN BASIC
AND US ES FILE NAMES TO LOAD FILES, BUT IT DOE SN'T HAVE A CATALOG SO YOU HAVE
TO CRAC K IT ANOTHER WAY BESIDES DEMUFFIN.HERE IS WHAT TO DO:

BOOT DUNGEON
WHEN IT SAYS"PLEASE WAIT" THEN PRESS RE
SET TWICE.
CALL-151
A44D:4C 69 FF
36:BD 9E 81 9E
MAXFILES1
CLOSE
LOAD HELLO
D6:0
  NOW YOU CAN GO TO BASIC AND LIST THE PROGRAM.BUT THERES MORE PROGRAMS TO THE
GAME!  SO WHAT YOU HAVE TO DO IS FIND TH E ENDING APPLESOFT ADDRESS AT $AF.B0
(L ISTED IN REVERSE ORDER) AND USE THE MON ITOR MOVE COMMAND TO MOVE IT INTO A
SAF E AREA.I CAN'T REMEBER THE ACTUAL ADDRE SS FOR THE PROGRAMS, BUT I WILL
GIVE YO U THE CORRECT FORMAT FOR DOING THIS:  6000<800.[WHAT EVER IS IN
$AB.F0,INREVE RSE ORDER]M [RETURN] THEN YOU BOOT DOS 3.3 AND MOVE IT BACK TO
THE CORRECT PLACE IN MEMORY:  800<6000.[6000+WHATEVER WAS IN AB.F0]M

  NOW FIX THE AB.FO TO WHAT THEY WERE BEF ORE AND SAVE THE PROGRAM!  IN AWHILE,
YOU WILL HAVE IT CRACKED!

BY THE WAY
D6:0 - CANCELS THE THING THAT MAKES THE
       PROGRAM IN MEMORY RUN EVERY TIME
       YOU TYPE A COMMAND IN APPLESOFT.

A44D:4C 69 FF - MAKES IT SO WHEN YOU LO
            AD AN APPLESOFT PROGRAM
            IT PUTS YOU IN THE MONI
            TOR.
36:BD 9E 81 9E RECONNECTS DOS

------------=> DOCTOR WHO <=-----------
```

```
==============================================================================
DOCUMENT deathcheat
==============================================================================
```

About "'Death Sword' - Cheat"

Let's face it, Death Sword is a great game...but... This cheat gives you
unlimited "hit-points". Warning: You can still get your head cut off and on
Game #2, after killing off 8 of those suckers, the big guy in Pink or Purple
(take your pick) will start whipping fireballs at you...try not to get hit..

Oh, and if you really feel like messing around with the number of hit points
your opponent starts with, its on Track 7,Sector 0,Byte $1E...

Later...

```
=============================================================================
DOCUMENT diskgo.txt
=============================================================================

-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-
           FAST DOS SPEEDS
-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-

TIMES ARE BASED ON THE TIME IT TAKES
TO BRUN THE BINARY FILE "FID"
(ACCURATE TO ABOUT 2/10 SECONDS)

DOS NAME      SPEED IN SECONDS
---------------------------------------
APPLE DOS 3.3     --> 7.5
DAVID'S DOS     --> 4.4
TURBO DOS   --> 2.8
DIVERSI DOS --> 2.5
FAST DOS    --> 2.6

IF YOU KNOW THE CORRECT TIMES FOR ANY
OTHER UNLISTED DOS PLEASE LEAVE FEED-
BACK AND IT'LL BE POSTED!

THANK YOU,

TAMERLANE OF THE RING

---------------------------------------

Enter (1-7, M=Menu, Q=Quit) :
```

===
DOCUMENT diskjock.app
===

EXAMINING PROTECTED APPLESOFT BASIC PROGRAMS.
BY THE DISK JOCKEY.

Many protected programs are written in APPLESOFT. Of course, most publishers are sly enough to protect against break ing out of their program with CTRL C or reset.   Also, most protect against re-entering BASIC from the monitor by changing the typical BASIC re-entry poi nt (at $3D0) so that it points to disaster. And lastly, many change the R UN flag vector at $D6 so if you manage to get out of their program and into BA SIC, anything you type will RUN their BASIC program.   I will describe how to b eat all these protection schemes, assuming you have an old style F8 monit or ROM.

First, we must determine if the protect ed program is written in APPLESOFT. If after you boot the progra m a BASIC prompt appears, this is a good indicator that at least some of the program is written in BASIC.   Further more, if the program prints a l ot of text on the screen, or requires a good deal of user inputs, it is a goo d guess that the program is written in BASIC. The reason for this is that p rinting text on the screen and inputing data from the keyboard is easi ly accomplished from BASIC using PRINT and INPUT statements. To do this from ASSEMBLY language requires a great deal more work. Also, we should relieze why a programmer uses ASSEMBLY language. The only real advantage to AS SEMBLER is speed. If speed is not critical, most (non-sadist) programmers will use BASIC.

With this in mind, look at how the prog ram runs and prints on the screen. If it runs at about the same speed as t he BASIC programs you have written run, it is a good guess that it is in B ASIC. Remember, ASSEMBLY language is considerably faster than BASIC in every respect.

Finally, read the package the program c ame in. It usually says what it was written it. If it doesn't, a dead give away is in the hardware requirements. If the program requires APPLESOFT in RO M, then at least part of the program is probably written in APPLESOFT.

Now that you have figured out your prot ected program is written in BASIC, it is time to LIST their code. The firs t step is to reset into the monitor when the program is running.

Now you can try to enter the immediate BASIC mode by typing:

```
*3D0G
```

This is the normal BASIC re-entry point . But if the protection is worth anything, this will not work.

Assuming that didn't work, reload the p rogram and reset into the monitor again.   The next thing is to try typing 9D84G or 9DBFG. These are the DOS cold
and warm start routines, respectively.   If you are lucky enough to get a BASIC prompt, you have done well. Most of the time, you won't.

If in either case you succeed in gettin g a BASIC prompt, try LISTing the program or CATALOGing the disk. If anyt hing you type starts the program running again, the protection has chang ed the RUN flag at $D6. So reset into the monitor again.

The RUN flag is a zero page location (a t $D6) which will run the BASIC program in memory if $D6 contains $80 o r greater (128 or greater in decimal). This is easy to defeat after you have r eset into the monitor by typing:

```
*D6:00
```

This resets the RUN flag to normal. Now if 3D0G, 9D84G or 9DBFG previously rewarded you with a BASIC prompt, this will solve the problem of the program re-running when you type a command.

For debugging efforts, the RUN flag can get changed from within a BASIC progra m by issuing the code:

```
10 POKE 214,255
```

or by poking location 214 with anything greater than 127. From ASSEMBLY language, the code would most likely lo ok like this:

```
800- A9 FF     LDA #$FF
802- 85 D6     STA $D6
```

or by loading a register with $80 or gr eater and storing it at $D6.

Now if 3D0G, 9D84G or 9DBFG did not pro duce a BASIC prompt, then the DOS being used is more elaborate. So re-loa d the program and reset into the monitor after it is running.

Now comes the final steps in trying to examine a BASIC program. If you are using a ROM card in slot zero with an o ld style F8 monitor ROM to reset into the monitor, turn on the mother board R OMs and turn off the ROM card INTEGER ROMs by typing:

```
*C081
```

Now reset the RUN flag to normal, just to be sure. Type:

```
*D6:00
```

Finally, enter APPLESOFT the sure fire way by typing:

```
*<CTRL C>
```

You should see an APPLESOFT prompt. Now type:

```
]LIST
```

and your APPLESOFT program should now a ppear.

Applying this to a real world example, try this method with one of Strategic Simulations releases (SSI). SSI uses a highly modified DOS called RDOS for their protection. SSI uses all the tric ks mentioned to prevent you from LISTing their programs. But using the a bove procedure, you can LIST their BASIC programs.

In addition, the DOS used by SSI (RDOS) uses the appersand in all of its DOS commands. So if you see any ampersands from within their BASIC program, you know it is a DOS command. For example, to catalog a SSI disk, after you follow the above procedure and you are in BASI C, type:

```
]&CAT
```

This will display SSI's catalog. Very d ifferent, eh!

Well, back at the ranch, if you want to save your APPLESOFT program to a norma l DOS disk, do these steps:

1) Reset into the monitor after the pro gram is running.

2) If you are using a ROM card in slot zero, Type:

```
*C081
```

3) Now type:

```
*D6:00
*9500<800.8FFM
```

3) Check where the APPLESOFT program en ds by typing:

```
*AF.B0
```

4) Write down the two bytes listed some where.

5) Boot a 48K normal DOS 3.3 slave disk with no HELLO program.

6) Enter the monitor by typing:

```
]CALL-151
```

7) Restore the APPLESOFT program by typ ing:

```
*800<9500.95FFM
*AF:  enter the two bytes you wrote down here, separated by spaces.
```

8) Enter BASIC and save the program by typing:

```
*3D0G
]SAVE PROGNAME
```

What you have done is to move $800 to $ 8FF out of the way so you can boot a slave disk. After normal DOS is up, you restore $800 to $8FF from $9500 to $95FF, and then restore the end of APPL ESOFT program pointers so DOS knows how big your BASIC program is. Next you just save it to your disk! Of course there are other more automated ways of getting programs to a normal DOS 3.3 disk (such as Demuffin Plus or CopyB), but this is a quick and dirty method that will always work. Keep in mind tha t the program may not run from normal DOS because of more secondary pr otection from within the BASIC program itself. Any curious CALLs, POKE s or PEEKs to memory above 40192 (this is memory where DOS resides) or b elow 256 (zero page memory) should be examined closely.

I hope this will help you learn more ab out the protected programs you own that are written in APPLESOFT.

COPYB DOCUMENTATION FILE. BY THE DISK JOCKEY.

INTRODUCTION:

There are probably hundreds of ways to protect a program from being copied. But generally speaking, protection fall s under two categories: protect the actual program (by various means), or p rotect a disk full of programs with some sort of DOS modification. DOS modi fications are the most common since they are the easiest to deal with (from the publisher's point of view). DOS modifications are also the least succes sful of protection, since someone always seems to find a way to copy all the files onto a normal DOS disk, eluding all the protection. The classic program for dealing with modified DOS' s is DEMUFFIN PLUS.   It works much the sam e way as Apple's MUFFIN program works. MUFFIN was written to read files from a DOS 3.2 disk and then write the m to a DOS 3.3 disk. DEMUFFIN was a varia tion of MUFFIN, allowing the hardcore 3.2 user to copy files from DO S 3.3 to DOS 3.2. DEMUFFIN PLUS operates on the same principle, but use s whatever DOS is in memory to read the disk, and then writes out to an ini tialized DOS 3.3 disk. While this is a powerful utility, it only works with programs that are based on DOS file structures and that have a catalog trac k.

INTRODUCING COPYB:

COPYB is a highly modified version of C OPYA which converts a protected disk that uses a modified DOS and/or RWTS to normal DOS 3.3 format.   The protected disk may have a normal DOS file structu re, or it may not. Since COPYB copies on a track by track basis, this does not matter. This makes COPYB a far more flexible tool than DEMUFFIN PLUS.

COPYB uses the protected disk's RWTS to read in the tracks and then uses norma l DOS 3.3 to write them back out to an in itialized disk. Unless otherwise instructed, COPYB copies track $03 to t rack $22, sector $0F to sector $00 of each track. Here are the parameters for COPYB:

```
LOCATION             NORMALLY
HEX DEC     DESCRIPTION     HEX DEC NT.
---------------------------------------
22E 558 FIRST TRACK TO READ   03 03 (1)
236 556 FIRST SECTOR TO READ  0F 15 (2)
365 869 RESET SECTOR NUMBER   0F 15 (2)
3A1 929 STOP ON ERROR($18=NO) 38 56 (3)
302 770 TRK TO STOP READING+1 23 35 (4)
35F 863 TRK TO STOP READING+1 23 35 (4)
```

NOTES (NT.):

1) This is the first track that COPYB s tarts reading at. This is normally set at track 3, so not to copy the protecte d DOS which normally resides on track 0 through track 2.

2) These two parameters are normally se t to $0F for 16 sector disks. Change these two parameters to $0C for 13 sect or disks. Most of today's protection schemes are based on 16 sect ors. Yet there are still a few using 13 sectors (such as Muse). Intere stingly enough, there is a handful of authors that also us sectori ng other than 13 or 16 sectors per track. An example of this is "Thief " from Datamost. This program uses 11 sectors per track. COPYB can al

so accommodate these programs.

at upon reading a 'bad sector' COPYB will stop and display an error. To let COPYB keep going after a read error, change this byte to $18 (24 in decimal) . The equivalent sector on the copied disk will be written blank.

4) These two parameter determine where COPYB will stop reading the protected disk. Normally, this is set to the last track, $22 (34 in decimal) , plus one. To change this, add one to the last tra ck you want to copy and change these two parameters.

CREATING COPYB:

After entering or downloading the BASIC program, save the program by typing:

]SAVE COPYB

Now you must enter the ASSEMBLY languag e subroutines that COPYB uses. COPYB uses the main subroutines that CO PYA uses, so we only have to modify the file COPY.OBJ0 that is on the DOS 3 .3 System Master. But first I will explain the added subroutines that COPY B needs.

Remember that COPYB uses the protected program's RWTS to read the disk by moving it from $8000 to $B700 - $BFFF. After COPYB is done reading the protected disk, normal RWTS is moved ba ck up to $B700 - $BFFF from $8900 to write to a normal DOS disk. This is han dled by some subroutines which will add to the existing file COPY.OBJ0. Her e are the routines (formatted in 80 columns):

```
  0220- 20 B0 02 JSR $02B0 :save the registers.
  0223- A0 B7 LDY #$B7 :botto m page to move from.
  0225- A9 89 LDA #$89 :desti nation page to move to.
  0227- 20 80 02 JSR $0280 :copy normal RWTS from $B700-BFFF to 89 00-91FF.
  022A- 20 B4 03 JSR $03B4 :subro utine to locate RWTS ($3E3).
  022D- A9 03 LDA #$03 :start ing track to read from.
  022F- 8D D1 02 STA $02D1 :store track.
  0232- 8D D2 02 STA $02D2 :store track.
  0235- A9 0F LDA #$0F :start ing sector to read from.
  0237- 8D D3 02 STA $02D3 :store sector.
  023A- 8D D4 02 STA $02D4 :store sector.
  023D- 4C E7 02 JMP $02E7 :jump to read routine.
  0240- 20 B0 02 JSR $02B0 :save the registers.
  0243- A0 80 LDY #$80 :botto m page to move from.
  0247- 20 80 02 JSR $0280 :move normal RWTS from $8900 back to $B700 -BFFF.
  024A- 4C F7 02 JMP $02F7 :jump to write routine.
  0260- 20 B0 02 JSR $02B0 :save the registers.
  0263- A0 89 LDY #$89 :botto m page to move from.
  0265- A9 B7 LDA #$B7 :desti nation page to move to.
  0267- 20 80 02 JSR $0280 :move normal RWTS from $8900 back to $B700 -BFFF.
  026A- 4C 17 03 JMP $0317 :jump to write routine
  0270- 20 B0 02 JSR $02B0 :save the registers.
  0273- A0 89 LDY #$89 :botto m page to move from.
  0275- A9 B7 LDA #$B7 :desti nation page to move to.
  0277- 20 80 02 JSR $0280 :move normal RWTS from $8900 to $B700 -BFFF.
  027A- 4C BC 03 JMP $03BC :Resto re the registers and exit.
  0280- 84 07 STY $07 :store original page to move from.
  0282- 85 09 STA $09 :store destination page to move to.
  0284- A2 09 LDX #$09 :load X with number of pages to move.
  0286- A9 00 LDA #$00 :load accum with $00.
  0288- A8 TAY :trans fer #$00 to Y.
  0289- 85 06 STA $06 :store #$00 at $06.
  028B- 85 08 STA $08 :store #$00 at $08.
  028D- B1 06 LDA ($06),Y:load accum with the address pointed to by locations
      $06 & $07 (lo-hi order), index ed by Y.
  028F- 91 08 STA ($08),Y:store accum at the address pointed to by locations $07
      & $08 (lo-hi order) index ed by Y.
  0291- C8 INY :incre ment Y.
  0292- D0 F9 BNE $028D :conti nue until end of page.
  0294- E6 07 INC $07 :incre ment original page.
  0296- E6 09 INC $09 :incre ment destination page.
  0298- CA DEX :decre ment X.
  0299- D0 F2 BNE $028D :if ha ven't moved 9 pages, do again.
  029B- 60 RTS :retur n from subroutine.
  02B0- 8D C7 03 STA $03C7 :store accumulator at $3C7.
  02B3- 8E C8 03 STX $03C8 :store X-register at $3C8.
  02B6- 8C C9 03 STY $03C9 :store Y-register at $3C9.
  02B9- 60 RTS :retur n from subrotine.
```

So to create the objective file for COP YB, we should first enter the monitor by typing:

```
]CALL-151
```

Next we should initialize the memory ar ea by typing:

```
*220:FF N 221<220.2CDM
```

Now bload the file COPY.OBJ0 from the D OS 3.3 System Master by typing:

```
*BLOAD COPY.OBJ0
```

Now type in the new code and some chang es:

```
*228:80 02 20 B4 03 A9 03 8D
*230:D1 02 8D D2 02 A9 0F 8D
*238:D3 02 8D D4 02 4C E7 02
*240:20 B0 02 A0 80 A9 B7 20
*248:80 02 4C F7 02
*260:20 B0 02 A0 89 A9 B7 20
*268:80 02 4C 17 03
*270:20 B0 02 A0 89 A9 B7 20
*278:80 02 4C BC 03
*280:84 07 85 09 A2 09 A9 00
*288:A8 85 06 85 08 B1 06 91
*290:08 C8 D0 F9 E6 07 E6 09
*298:CA D0 F2 60
*2B0:8D C7 03 8E C8 03 8C C9
*2B8:03 60
*2C1:20
*2C4:40
*2C7:60 02
*2CB:13 7F B0 60
*2D0:01 03 03 0F 0F
*2D8:B4
*2DD:02
*2F8:B4
```

```
*318:B4
*3C7:02 9D C0 B3 C4 C4
*220:20 B0 02 A0 B7 A9 89 20
```

After entering these changes, save the file by typing:

```
*BSAVE COPYB.OBJ,A$220,L$1AB
```

USING COPYB:

To use COPYB, you must capture the fore ign RWTS and put it at locations $8000 through $88FF.   You can do this on e of two ways:

1) Boot the protected disk and after th e foreign DOS is loaded, reset into the monitor. The foreign DOS will usual ly be loaded a few seconds after the boot starts. You can tell this beca use many times a BASIC prompt will appear at the bottom of the text screen . Use the monitor move command to move RWTS down to $8000 as so:

```
  *8000<B700.BFFFM
```

Now boot a 48k slave disk (this will no t destroy memory from $900 to $95FF) and run COPYB.

ENTERING THE PARAMETERS AND RUNNING COPYB:

Run COPYB by typing:

```
]RUN COPYB
```

The program will come up and ask what p arameters to use, all described above. COPYB will poke in the values you have entered for you. Enter all values in DECIMAL.

After entering the parameters, you will be asked if your selections are correct. If you answer YES, the next se t of prompts will appear, which should look familiar.   Enter the origina l and destination drive and slot numbers, just like in COPYA.   Lastly, yo u will be asked if you want the destination disk to be initialized, res pond yes or no. Now press the RETURN key to start the copy.

When the copy is completed, assuming al l went correctly, you will have a normal DOS 3.3 version of your protecte d disk which may run or be examined and changed more easily then t he original disk.

This method of deprotection is more dep endable that using DEMUFFIN PLUS and covers more types of programs.   I am sur e you will find COPYB an excellant utility to have.

INTRODUCTION TO KRAKING PART TWO.B. MAK ING YOUR OWN CUSTOM F8 MONITOR ROM. BY THE DISK JOCKEY.

In this section I will describe how to make the code for the modified F8 monitor ROM that you will find extremel y useful in kraking.

The EPROM will act like a old style F8 monitor ROM with regards to resets. What I mean is that hitting reset will cause you to jump into the monitor.

The EPROM will also have a special func tion when an NMI is encountered. Upon NMI, this ROM will push the accumu lator, the x-register,the y-register and location $00 onto the st ack. The stack pointer will then be saved at location $00.

Next, the EPROM will move $00 to $4000 into a RAM card in slot zero. This clears the way for a 16K slave disk boo t. Here is the code and an explanation of how it works (in 80 colu mn format):

```
FCC9- 48        PHA       PUSH
ACCUM ONTO THE STACK.
FCCA- 8A        TXA       TRANS
FER X-REG TO ACCUM.
FCCB- 48        PHA       PUSH
(X) ACCUM ONTO THE STACK.
FCCC- 98        TYA       TRANS
FER Y-REG TO ACCUM.
FCCD- 48        PHA       PUSH
(Y) ACCUM ONTO THE STACK.
FCCE- A5 00     LDA   $00       LOAD
ACCUM WITH $00.
FCD0- 48        PHA       PUSH
($00) ACCUM ONTO THE STACK.
FCD1- BA        TSX       TRANS
FER STACK POINTER TO X-REG.
FCD2- 86 00     STX   $00       STORE
 STACK POINTER AT $00.
FCD4- AD 81 C0    LDA   $C081   ENABL
E WRITE TO RAM BANK 1.
FCD7- AD 81 C0    LDA   $C081   (MUST
 ACCESS TWICE).
FCDA- A0 00     LDY   #$00      -----
------------------------------
FCDC- B9 00 00    LDA   $0000,Y MOVE
$00 TO $FF INTO RAM CARD SO WE
FCDF- 99 00 D0    STA   $D000,Y CAN U
SE ZERO PAGE FOR REST OF MOVE.
FCE2- C8        INY
FCE3- D0 F7     BNE   $FCDC   -----
------------------------------
FCE5- 84 00     STY   $00       MOVE
$100-$2FFF INTO BANK 1
FCE7- 84 02     STY   $02       OF TH
E RAM CARD.
FCE9- A9 01     LDA   #$01
FCEB- 85 01     STA   $01
FCED- A9 D1     LDA   #$D1
FCEF- 85 03     STA   $03
FCF1- B1 00     LDA   ($00),Y
FCF3- 91 02     STA   ($02),Y
FCF5- C8        INY
FCF6- D0 F9     BNE   $FCF1
FCF8- E6 03     INC   $03
FCFA- E6 01     INC   $01
FCFC- A5 01     LDA   $01
FCFE- C9 30     CMP   #$30
FD00- D0 EF     BNE   $FCF1   -----
------------------------------
```

```
FD02- 4C CD FE    JMP   $FECD   RAN O
UT OF ROOM HERE, JMP TO $FECD.
.
.
.
FECD- A9 D0     LDA   #$D0      RESET
 MOVE ROUTINE POINTERS.
FECF- 85 03     STA   $03
FED1- AD 89 C0    LDA   $C089   ENABL
E BANK 2 OF RAM CARD.
FED4- AD 89 C0    LDA   $C089   (MUST
 ACCESS TWICE).
FED7- B1 00     LDA   ($00),Y -----
------------------------------
FED9- 91 02     STA   ($02),Y MOVE
$3000-$3FFF INTO BANK 2
FEDB- C8        INY       OF TH
E RAM CARD.
FEDC- D0 F9     BNE   $FED7
FEDE- E6 03     INC   $03
FEE0- E6 01     INC   $01
FEE2- A5 01     LDA   $01
FEE4- C9 40     CMP   #$40
FEE6- D0 EF     BNE   $FED7   -----
------------------------------
FEE8- AD 82 C0    LDA   $C082   TURN
ON MOTHERBOARD RAM AND WRITE
FEEB- AD 8A C0    LDA   $C08A   PROTE
CT BANKS 1&2 OF RAM CARD.
FEEE- 4C FD FE    JMP   $FEFD   RAN O
UT OF ROOM, JUMP TO $FEFD.
.
.
.
FEFD- A2 1C     LDX   #$1C      THIS
SUBROUTINE OUPUTS THE
FEFF- BD 0B FF    LDA   $FF0B,X MESSA
GE "RAM CARD LOADED WITH
FF02- 9D D6 07    STA   $07D6,X $00-3
FFF" AT THE BOTTOM OF
FF05- CA        DEX       THE T
EXT SCREEN.
FF06- 10 F7     BPL   $FEFF   -----
-----------------------------
FF08- 4C 59 FF    JMP   $FF59   ALL D
ONE, EXIT THRU NORMAL RESET.
```

To create this EPROM file, here are the steps:

1) Boot a normal DOS disk and enter the monitor by typing:

```
]CALL -151
```

2) Move your autostart F8 monitor ROM c ode down into RAM by typing:

```
*4800<F800.FFFFM
```

3) Now change the code as follows:

```
  *4CC9:48 8A 48 98 48 A5 00 *4CD0:48 BA 86 00 AD 81 C0 AD 81 C0 A0 00 B9 00 00
99 *4CE0:00 D0 C8 D0 F7 84 00 84 02 A9 01 85 01 A9 D1 85 *4CF0:03 B1 00 91 02 C8
D0 F9 E6 03 E6 01 A5 01 C9 30 *4D00:D0 EF 4C CD FE *4ECD:A9 D0 85 03 AD 89 C0 AD
89 C0 B1 00 91 02 C8 D0 F9 E6 03 *4EE0:E6 01 A5 01 C9 40 D0 EF AD 82 C0 AD 8A C0
4C FD FE *4EFD:A2 1C BD 0B FF 9D D6 07 CA 10 F7 4C 59 FF *4F0B:52 41 4D 60 43 41
52 44 60 4C 4F 41 44 45 44 60 57 49 54 48 60 *4F20:64 70 70 6D 73 46 46 46
*4FFA:C9 FC 59 FF
```

4) Now save the file to a disk by typin g:

```
*BSAVE F8 SAVE RAM EPROM,A$4800,L$800
```

5) Finally, burn the 2716 EPROM with th is code or have someone do it for you.

Now to use your new 2716 EPROM, you mus t make these changes directly to the chip itself (not advisable), or to a ju mper socket which your new chip will plug into, and then which will be plugg ed into your motherboard.

You need a 24 pin low-profile socket (n ot wire-wrap!, they will destroy your motherboard sockets!). These are availa ble from radio shack (part number 276-1989) or the such. Now with the soc ket up-side-down and the pins looking you in the face, it should look like this:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ !
!                              !
!                              !
!                             -
!                    (notch)->!
!                             -!
!                              !
! .  .      .  .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------
```

Now your soldering skills come in handy ! Using some short, hi-gauge wire (wire-wrap is preferable, but anything in the 26-30 gauge will work), solder a piece between pins 21 and 24, and solde r a piece between pins 12 and 18. Be extremely careful not to short out t he wire or to cross solder any pins! Also, try and solder as close to the ba se of the socket as possible, since you have to cut off pins 18 and 21 afte r you have finished soldering them. Now cut of pins 18 and 21 as close to t he base of the socket without cutting the freshly soldered wires! Remember, p ins 18 and 21 should be short enough so that they will not touch the socket you will be plugging this one into. The socket should now look like this:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ !
!           x     x      /  !
!          /      /      /   !
!           /    /--------/   -
!    /--------/            !
!   /                     -!6
!  /                       !
! /  .      .  .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------
```

Double check your soldering and the con nections (and notice that pin 18 and 21 are cut off!). Now carefully remove the ROM labelled F8 (it is the socket farthest on the left that has 24 pins a s you face the keyboard) and plug this jumper socket into the motherboard. Now plug your modified 2716 EPROM into thi s jumper socket and your all done! Make s ure you have the notch pointing in the same direction as the other ROMs (towar ds the keyboard).

When you turn on the Apple you should s ee a screen full of garbage with the monitor prompt at the bottom of the scr een. To boot your Apple, just type "6 ctrlP", and your computer will act j ust as usual.

INTRODUCTION TO KRAKING PART TWO.C. MAKING YOUR OWN NMI BOARD. BY THE DISK JOCKEY.

In this article I will describe how to make your own NMI board that will work in conjunction with the modified F8 monitor EPROM you have (or will) create. Here is the parts lists for the NMI board:

(1) 7400 or 74LS00 chip. Radio Shack part #276-1801. $0.59

(1) SPDT momentary push switch. Radio Shack part #275-1549. $2.19

(1) 14 pin low profile or wire wrap socket. Radio Shack part #276-1999 or #276-1993. $0.89

(2) 3.3k ohm resisters, 1/4 watt. Radio Shack part #271-1328. $0.39

(1) Dual plug-in interface board. Radio Shack part #276-164. $4.95 NOTE: This part has been discontinued by Radio Shack, but you can sometimes still find them in the junk bin. Cut the board so it will fit inside your Apple.

ALTERNATIVELY: (1) Apple bare board number PAPGBP5001. $13.95 from Priority Electronics, 9161 Deering Ave., Chatsworth, CA 91311.

After you have obtained all the parts above, you should solder the 14 pin socket and the two resistors somewhere convenient on the bare board. Next get some hi-gauge wire and make the following connections:

1) Connect pin 25 of the bare board to one leg of each of the two resistors.

2) Connect each of the other legs of the resistors to the two outside contacts of the switch. (one resistor goes to one contact, the other resistor goes to the other contact). Use some good wire.

3) Connect pin 25 of the bare board to pin 14 of the 14 pin socket.

4) Connect the middle contact of the switch to pin 7 of the 14 pin socket.

5) Connect pin 26 of the bare board to pin 7 of the 14 pin socket.

6) Connect pin 2 of the 14 pin socket to pin 6 of the 14 pin socket.

7) Connect pin 3 of the 14 pin socket to pin 4 of the 14 pin socket.

8) Connect pin 4 of the 14 pin socket to pin 29 of the bare board.

9) Connect the leg of one of the resistors that is connected to the switch to pin 5 of the 14 pin socket.

10) Connect the other leg of the resistor that is connected to the switch to pin 1 of the 14 pin socket.

11) Check all connections twice, and don't get confused on what pin is what on the bare board and the 14 pin socket.

You have now completed building your own NMI board. This board may be plugged into any one of the peripheral slots.

THE ROM CARD. BY THE DISK JOCKEY.

OBJECTIVE:

While being able to deprotect programs from files on Pirate's Harbor is certainly helpful in the quest for copyable software, it would be optimal to deprotect your own programs without the help of other sources. This works on the premise that you can give a man a fish and feed him today, or you can teach him to fish and feed him for life.

With this in mind, our objective is to teach you the ways of deprotection in general, and mention some of the tools that will make this easier. Although some of these tools will cost money and are somewhat specialized, they will only increase your knowledge of the Apple computer, for what that's worth.

INTRODUCING THE ROM CARD:

The foremost of important tools for easily snooping through memory is the ROM card. The ROM card was originally developed for t`g{m(a|`(xzograms written in both INTEGER and APPLESOFT BASIC. Remember that your motherboard (the big green printed circuit board inside your computer case) can house only one of the BASIC languages, either INTEGER or APPLESOFT. When the Apple was originally released, it was only available with INTEGER BASIC. So many programs were written in INTEGER, and would not run on the Apple II+ (with APPLESOFT on board) when it was introduced.

Before RAM memory was very cheap, many people bought ROM cards for their Apple II+ that could be put in slot zero (as you would a RAM card), to enable them to run programs that were written in either BASIC language. It was just as though you had loaded INTEGER BASIC into you RAM card, like the DOS 3.3 System Master's HELLO program does. When RAM cards became available at a reasonable cost, everyone started buying them because they are so much more versatile for the average folk. That is why you don't see ROM cards for sale too much any more. But for deprotecting Apple programs, the ROM card is indespensible.

Also, for the Apple II owner who wanted to run the newer APPLESOFT programs, the ROM card was available with APPLESOFT ROMs. The INTEGER and the APPLESOFT versions of the ROM card are identical, except for the actual ROMS on the card. In other words, one had INTEGER ROMs and the other had APPLESOFT ROMs, and there is no other differences.

THE REASONS WHY:

Their are several reasons the ROM card is so important. The least of the reasons is the need for INTEGER BASIC or the Programmer's Aid chip. If you can get a ROM card cheaply without INTEGER or the Programmer's Aid ROMs, do so. From a cost outlook, it is to your advantage. Besides, INTEGER is a dead dinosaur, and who really cares if it's faster than APPLESOFT?

The reason we want a ROM card is so we can put an old style F8 monitor ROM and THE INSPECTOR ROM (from Omega Microware) on it. These two ROM chips are really essential for learning more about protected programs. Ultimately, we would like WATSON in conjunction with THE INSPECTOR, but to do so you will also need INTEGER BASIC ROMs, since WATSON uses some routines from the INTEGER BASIC ROMs. Watson enhances the Inspector by adding even more commands and flexibility. The combination of Watson and the Inspector provides you with great power for not only snooping, but also for general purpose utility chores.

The reason we want the old style F8 ROM should be obvious by now. After reading several kraking articles and from your own experiences, you have noticed that it is impossible to break out of many programs with just an autostart F8 monitor ROM. The reason we should have the old style F8 ROM on the ROM card and not on the mother board is for convenience. The ROM card has a switch which determines which F8 monitor ROM is active when you hit reset. So you can have the convenience of the Autostart F8 monitor ROM, and when you need it, hit the switch and be able to break out of any program you want with the old style F8 monitor ROM.

OBTAINING YOUR OWN ROM CARD:

ROM cards are available used at very cheap prices. Check your local Apple users' group. Alternatively, you can get blank cards and stuff it yourse8f. I would suggest stuff your own since the parts are easy to get, and it is usually the least expensive route! I have also seen Japanese clone cards for sale at a very reasonable price. The best place to check for these is in The Computer Shopper, a bi-monthly newspaper of Apple and other computer bargins.

OBTAINING YOUR OWN ROMs:

You can either buy an old style F8 monitor ROM, or you can make one by changing your autostart F8 code slightly. After making the change, you can save the file to disk and have a friend or your local computer store burn the image into a 2716 EPROM. Here is the instructions for creating your own:

1) Boot a normal DOS 3.3 disk. 2) Enter the monitor by typing:

```
]CALL-151
```

3) Move the autostart F8 ROM image into RAM by typing:

```
*4800<F800.FFFFM
```

4) To enter the monitor when reset is pressed, type these changes:

```
*4FFC:59 FF
```

5) Bsave the file to a blank disk by

typing:

```
*BSAVE OLD $F8,A$4800,L$800
```

6) Burn this image into a 2716 EPROM.

This new F8 EPROM will be just like the autostart version F8 ROM except when you hit reset, you will be in the monitor and not in BASIC. Now you can reset out of any program.

Alternatively, you can use a modified F8 EPROM too, as described in other kraking articles. This will give you the advantage of being able to save memory from $00 to $8FF when you hit reset. This would certainly be helpful at times.

If you want INTEGER BASIC on your ROM board, you can either buy the ROMs from your local Apple dealer, or you can make them. When you bought your Apple disk drive and controller you also bought DOS 3.3, the DOS 3.3 System Master, and all the programs on the System Master, including INTEGER BASIC. So you can also burn INTEGER into 2716 EPROMs just like you burned your new F8 EPROM, and put them on your ROM card. Here are the steps to do this:

1) Boot your DOS 3.3 System Master.

2) Bload the file INTBASIC by typing:

```
  ]BLOAD INTBASIC,A$2000
```

3) Bsave the INTEGER files to a blank disk by typing:

```
]BSAVE INT $E0,A$3000,L$800
]BSAVE INT $E8,A$3800,L$800
]BSAVE INT $F0,A$4000,L$800
```

4) Burn three 2716 EPROMs from each of these files.

IMPORTANT: In order to use 2716 EPROMs on your ROM card instead of the F8 ROM socket on the ROM board white circle with the word "2716" next to it. Inside the circle will be four solder pads, grouped into two pairs. Notice each pair has two pads real close together, but not touching. Take a soldering iron and cross each pad in each pair together with some solder. So now the circle will have two solder pads, instead of four. DO NOT CROSS ALL FOUR PADS TOGETHER! Your ROM board will now except ONLY 2716 EPROMs, so when you do this you have to use all 2716 EPROMs, and no 9316 ROMs.

While on the subject of jumpers, there is another jumper on your ROM card just below the E8 ROM. This jumper, when crossed, will ignore the position of the ROM card switch. Reset will always ignore the F8 monitor ROM on the ROM board, and just use the motherboard F8 monitor ROM. Obviously, we do not want to cross this jumper.

If you can't tell if you should cross the 2716 jumper because you don't know if you have 2716's or 9316's, it is easy to tell the difference. 2716's have a small quartz window on their face, usually beneath some label.   The window is used to erase the EPROM (hence the name Erasable, Programable,0Read Only Memory). They should also say "2716" somewhere on them too.

If you must mix 9316's and 2716's on the same ROM card, do not cross any of the two pairs of jumpers. Instead, refer to INTRODUCTION TO KRAKING PART 2.B on how to make 2716 scrambler sockets for using 2716's in 9316 applications.

9316's are the all black 24 pin ROM chips that come with your Apple, and are not erasable. They will not have a quartz window.

Now plug in your F8 monitor EPROM or ROM in the socket labeled F8, and do the same with the other E0, E8 and F0 INTEGER EPROMs or ROMs. We are ready for the next step.

THE INSPECTOR:

The next thing the ROM board enables us to do is to use THE INSPECTOR from Omega Microware. The Inspector is basically a sector editor program with some really nice features which come in handy when deprotecting programs. To use The Inspector, we just reset out of a program and into the monitor, and type C080 N D800G. Now The Inspector is running without disturbing anything in memory outside of what normally gets disturbed upon hitting reset.

Besides being a sector editor, The Inspector has a very useful FIND command which enables us to find any string of bytes in memory or to locate them on a disk. This can help us find where a particular routine is being called from, or to help find the starting address of a program, etc. Also, The Inspector has a free sector map, removes DOS from a disk, does nibble reads of protected disks, displays bytes in HEX or ASCII, reads half tracks, and compares or verifies disks. It also has unlimited uses in snooping and changing memory and disks.

The Inspector is VERY useful, especially in conjunction with its partner, WATSON (also from Omega1Microware). It is the most powerful and well used utility I have. And since it is on my ROM card, it is always available without disturbing mother board memory. This is why it is so useful. If we had to load it in from disk like any other program, it would be just like any other sector editor to a large extent.

Ask around and try and find someone with the Inspector and Watson code saved in a Bfile so you can burn your own Inspector EPROM and plug it into your ROM card. If you buy the Inspector, BE SURE you tell Omega when buying The Inspector that you want it in 2716 EPROM form if you are planning on using only 2716 EPROMs on your ROM card, instead of 9316's.

WHERE Do I PUT IT?:

Now that you have a ROM board, what slot should you put it in? Generally, the conventional slot is slot zero. But, I am sure many of you have RAM cards in slot zero. It is probably best 99 percent of the time to have your RAM card in slot zero, since most programs which use RAM cards expect it in only slot zero (although it has some uses in other slots). So that leaves you with two choices, put your ROM card in another slot, or play musical slots when you need the ROM card.

I prefer to put my ROM card in slot two since the card (and The Inspector) is still always available, but that presents some problems. The main problem is that after flipping the ROM board switch up to use the old F8 monitor ROM and hitting reset, your computer cannot find APPLESOFT when you boot a disk, it can only find INTEGER BASIC (assuming you have it on the ROM card). One way out is to flip the switch back down and hit reset again before booting a disk. I do not recommend this when deprotecting a program since now your computer will jump to the reset routine that was there when you originally hit reset. Of course, there is a better way.

After reseting into the monitor and just before you boot a disk you must turn off your ROM card ROMs and turn on the motherboard ROMs. This is accomplished with a softswitch, much like turning on the hi-res page. Remember how we activated the Inspector with C080 N D800G? Well, the C080 turns on the ROM card, so those ROMs are active, much like typing INT from BASIC. If you type C081 from the monitor, this turns the ROM card ROMs off, and the motherboard ROMs on. If your ROM card is in another slot, you need to type the slot number times ten, and add it to C081. Then you can boot a disk, and APPLESOFT will be
found. Here is a chart of what you would type from the monitor just prior to booting a disk (you do not have to do this if your ROM card is in slot zero):

SLOT	TURN ON ROM CARD	TURN ON MOTHERBOARD
0	C080	C081
1	C090	C091
2	C0A0	C0A1
3	C0B0	C0B1
4	C0C0	C0C1
5	C0D0	C0D1
6	C0E0	C0E1
7	C0F0	C0F1

For example, if your ROM card was in slot two, and you have reseted into the monitor, type:

```
*C0A1
```

before you boot a disk to turn on your motherboard ROMs so APPLESOFT can be found.

Likewise, if you have reset into the monitor and you want to use the Inspector, type (assuming slot two):

```
*C0A0 N D800G
```

Notice we multiply the slot number by twenty and add it to $C080 or $C081.

Another alternative is to use DAVID DOS from David Data when you boot normal DOS 3.3. This DOS is incredible in just speed savings of loading programs. It will also recognize your ROM card in any slot (and hence solves our problem), has a relocatable DOS function to put DOS in your RAM card, has a find command, and has a disassemble command. If that is not enough, it has a TLOAD and TLIST
command which loads and lists text files like BASIC or binary files! This alone make DAVID DOS worth the price. The only disadvantage to David DOS is it does not have an INIT disk command. To put David DOS on another disk requires using
a program that comes with it.

Of course, if you are booting a disk which does not run under normal DOS, you can not use David DOS and you must use the first alternative.

CONCLUSION:

This completes our discussion of ROM cards and what configuration is most desirable. In summary, we would like a ROM card with an old style F8 monitor ROM, The Inspector, and ultimately, INTEGER BASIC and WATSON. Next we will discuss some general methods of deprotecting single load programs.

INTRODUCTION TO KRAKING PART TWO. USING
SOME MINIMAL HARDWARE.
BY THE DISK JOCKEY.

INTRODUCTION:

Assuming that you have read part one of this series, you now should possess some basic information regarding the ar chitecture of the Apple computer. Using this basic information you will g o quite far down "memory lane" in your kraking efforts, but it doesn't stop th ere. Now we need to talk about some basic hardware you will need to make yo ur job easier. What we will be discussing is the use of resets and "NM Is" in the art of kraking.

As you have probably noticed, when you try to reset from a protected program with your II+ or //e, the computer can do some strange things. This is because the reset key is actually a programable key that when hit, can be made to run a program within memory. In most cases, t he program that is run clears memory and re-boots your disk. This of course keeps undesirables from snooping through memory, discovering any secrets a publisher may be hiding.

The reason the reset key is programable computer will jump to the address point $3F3, in "backassward" order. This mean s if $3F2 = 00 and $3F3 = 60, then upon reset you will jump to $6000. The worst part about this is there is no way to stop it unless you use some hard ware (although you may use a RAM card, I will discuss this method later) .

The hardware I am getting to is the fam ed "old F8 monitor ROM", which when you hit reset, jumps unconditionally to $FF59 and puts your program to a halt and leaves you in monitor. Using this c hip, you may break out of any program and examine memory. Now you ask, "what the hell is a old F8 monitor ROM anyways?".

The F8 monitor ROM is a set of programs that oversees the operations of your Apple, and hence is called a "monitor". It is a ROM because it is "Read Only Memory", or a permanent memory, as oppo sed to random Access Memory, or RAM. The reason it is called a "F8" ROM is because it occupies memory from $F800 to $FFFF. The chip is located jus t in front of the peripheral slots on the II+, and should be labeled "ROM- F8". On a //e, this chip is not as accessible as on the II+, and generally you are "SOL" (shit out of luck) in trying to replace it. But fear not, ref er to the article "The ROM card" or "RAM card Resets" for help in your effo rts.

Back in the old days when the Apple was first introduced, it came with the "ol d style F8 monitor ROM". But later it was replaced by the "autostart F8 monitor ROM". It would be most easy for us to f ind the old style ROM and replace it with our present autostart monitor ROM. This would allow us to reset out of an y program, at any time with it installed. But before you run out and buy one, read on as I introduce another topic th at will parallel our ROM discussion.

INTRODUCING THE NMI:

NMI is an acronym for NON MASKABLE INTE RRUPT, and as the name implies, it can not be prevented (or masked) on the Apple. The NMI is the basis behind mos t of the "copy cards" on the market, such as the Wildcard or Replay cards. The NMI allows us to interrupt a program, a nd to restart it with minimal effort. Obviously this is of extreme importance to the krakist, who wants to interrupt a programs, save memory to a normal DOS disk, and restart the program upon BRUNing the file.

To use an NMI you can simply crossed pi n 26 (ground) and pin 29 (NMI) of any one of the peripheral slots. You can do this with a 100 ohm resistor. This wil l execute an NMI.

Unfortunately, this is less than ideal since when you try to do this, you will probably execute 20 or so NMI's. This i s because it happens so fast, that an NMI will interrupt an NMI, and so on fo r many, many times. This will put much garbage onto the stack (page one). Using a switch for this chore doesn't help since the switch actually slams (o r bounces) against itself many times causing the same problem. To solve this we need to make a "de-bounced" NMI switch. This will constitute about $8 t o $20 of capital resources (depending on your parts supplier), and a solderin g iron. This is considerably less expensive than a store bought NMI board, but will lack some of the features the commercial ones have. A fu ll discussion of how to make an NMI board is in the file "KRAKING PART TWO. C".

Assuming you have made your NMI card, I will now tell you more about how it works and its uses. If you don't fully understand the workings of the NMI, don't worry about it. Just try and foll ow along.

When you push the NMI switch, the 6502 processor will push the present value of the program counter on the stack alo ng with the processor status word. Then it will jump to what ever location s are pointed to by $FFFA and $FFFB. So the restart a interrupted pro gram, we only need to restore the registers (x, y, accum), the lower page s of memory, and the stack pointer, and do a "RTI" (return for int errupt) instruction.

Now remember our old F8 monitor ROM? We ll these two locations live in the monitor ROM. It would be nice if we cou ld change these location and after an NMI is executed, run a small program to that will save the registers, the stack pointer, and the lower pages of m emory. Now this leads us back to our old friend, the F8 monitor ROM.

This is indeed what we need to do. The best thing would be to execute an NMI, and then jump to a routine that moves t he lower 16K of memory into a RAM card. Then we could boot a 16K slave di sk (which would only disturb the lower 16K of memory), and save all of m emory to a disk. After we have saved all of memory, we could reconstruct our program and re-start, or do a "return from interrupt".

Of course to do this we need to change some of the code in the F8 monitor ROM. We can not do this directly to the F8 c hip that comes with your Apple since it is Read Only Memory. But we can move the code in the ROM down to RAM, put our routines in, and burn a new "2716 E PROM". The 2716 EPROM will replace the ROM, and will have our new kraking rout ines in it.

Now you ask, "how do I burn a 2716 EPRO M?". Well, if you don't have access to an EPROM programmer, you can take yo ur modified F8 code (saved to a disk) to a local computer store and they shou ld be able to burn you one for a nominal fee.

Refer to the article entitled "KRAKING PART TWO.B" for an explanation of how to create the code for the new EMPROM a nd how to plug it in after it is burnt.

Lastly, we need to make a 16K slave dis k and to use the program to save all of memory to a disk. To get the program type it in or download it from someone . To create a 16K slave disk, do the foll owing: (NOTE: this only applies to the Apple II or II+)

1) Turn off your computer.

2) Open the lid, and look for the 3 row s of chips that have a white line boarder around them. These are the 48K of RAM in your Apple II+.

3) Remove any one chip from each of the two rows of RAM furthest away from the keyboard.

4) Turn the computer on and boot your D OS 3.3 System Master.

5) Put a blank disk in the drive and ty pe:

```
]INIT RAM 48K SAVER
```

6) When this is complete, turn the comp

uter off and replace the two chips.

7) Run a sector editor and change the f ollowing sectors of the 16K slave disk:

TRK	SECTOR	BYTE	FROM	TO
$00	$01	$48	$03	$00
$00	$0D	$42	$06	$34

8) Write the sector back out to your 16 K slave disk.

9) Delete the Hello program on the disk by typing:

```
]DELETE RAM 48K SAVER
```

Now download the "RAM 48K SAVER" file a nd save it to your 16K slave disk. Also down load the file "MEMORY MOVE WR ITER". Save these to your 16K slave disk also, and then write protect it.

In the next episode, I will discuss how to use these hardware and software in a real-life application.

DEPROTECTION PART THREE.
PRACTICAL USES FOR THE NMI/MODIFIED ROM HARDWARE.
BY THE DISK JOCKEY

Now that you have burned your own F8 mo nitor ROM, constructed your own NMI board and created a 16K slave disk with the previously mentioned files, its time to put it all together and use it (also make sure you have a RAM card in slot 0). The primary use for these hard ware devices is for the single load program. As a practical example, we wil l be putting the Locksmith 5.0 fastcopy program into a file. This prog ram is a really fast normal DOS copy program that is

worth having in a file.

First turn off your computer and instal l your new F8 monitor EPROM into the motherboard, and put your NMI board in any slot. Now boot your Locksmith 5.0 (an original or a copy will do) and select the "16 sctr utilities" option. Next select the "16 sector fast disk ba ckup". Now just after the drive stops spinning, and before you see the prompt "drive- original:1", hit the NMI switc h on your NMI card. You should then be in the monitor.

Now boot your 16K slave disk. The "RAM 48K SAVER" program will run and will initialize a disk and save all 48K of m emory to your disk.

Finally, run the "MEMORY MOVE WRITER" p rogram and select the number of moves as one. Next select the running address as $8000. Use a forward memory move, and enter the start page as $40, and th e hi page as $80. Next select the starting page to move to as $00. Finall y, select the text page, page one, and full text. Now enter $8024 as the address to jump to and save the memory move program to disk.

Now its time to put all these files tog ether as the final product. Boot a normal 48K disk and Bload the following files by typing:

```
]BLOAD ^00-3FFF,A$4000
]BLOAD MEMORY MOVE $8000,A$8000
]BLOAD RERUN,A$8024
```

Now make the file run when you brun it by typing:

```
]CALL -151

*3FFD:4C 00 80
```

Now we can save the final product by ty ping:

```
*BSAVE LS 5.0 FASTCOPY,A$3FFD,L$4040
```

Congratulations! You now have deprotect ed the Locksmith fast copy program into a file that you may brun a nytime!

This technique will work well for depro tection other single load programs too! The main advantage to this techniq ue is that you don't have to find the starting address of the progra m to restart it. The program will just start up from the point where you interrupted it.

The only other thing you really must do is determine what parts of memory you must save so the program will run. REME MBER, YOU MUST ALWAYS SAVE MEMORY FROM $00 TO $2FF FOR THIS PROCESS TO WO RK CORRECTLY! Use the Memory Move Writer to rearrange memory so you can s ave it in a normal DOS binary file.

If you want further practice in using y our NMI/F8 EPROM hardware, write a program in APPLESOFT that some some scr een displaying and interrupt the program. Then try and reconstruct it us ing the same technique as described above and restart the program.

You can save the BASIC program in a Bfi le by saving $00 to $7FF and from $800 to the end of the program, where e ver that might be (zero page locations $AF and $B0 will give you the ending lo cation of a APPLESOFT program, in backassward order). You might also have to save some of the variable storage

for your BASIC program, which lives fro m $95FF down (depending on size). The best thing to do is to experiment, and practice makes perfect.

```
================================================================================
DOCUMENT dos.chart
================================================================================

DOS 3.3     READ        ###       WRITE         ###

START     B955  D5      ___     BC7A  D5        ___
 OF       B95F  AA      ___     BC7F  AA        ___
ADDRESS   B96A  96      ___     BC84  96        ___

END OF    B991  DE      ___     BCAE  DE        ___
DATA      B99B  AA      ___     BCB3  AA        ___

        SYNC FOR ADDRESS MARK   BC60  FF        ___

START     B8E7  D5      ___     B853  D5        ___
 OF       B8F1  AA      ___     B858  AA        ___
DATA      B8FC  AD      ___     B85D  AD        ___

END OF    B935  DE      ___     B89E  DE        ___
DATA      B93F  AA      ___     B8A3  AA        ___

        SYNC FOR DATA MARK      B83E  FF        ___

      ADDRESS          DESCRIPTION

   -21933 AA53,AA54    CHAR OUT VECTOR
   -21931 AA55,AA56    CHAR IN VECTOR
   -21929 AA57         CURRENT MAXFILES
   -21928 AA58         DEFAULT MAXFILES
   -21914 AA66,AA67    VOLUME (READ)
   -21912 AA68,AA69    DRIVE
   -21910 AA6A,AA6B    SLOT
   -21908 AA6C,AA6D    RECORD LENGTH
   -21906 AA6E,AA6F    RECORD NUMBER
   -21904 AA70,AA71    BYTES IN RECORD
   -21902 AA72,AA73    ADDRESS
   -21899 AA75-AA93    FILENAME($0=CAT)
   -21834 AAB6         WHICH BASIC (0 = INT; 64 =ASOFT ROM; 128 =ASOFT RAM)
          AC01         CATALOG TRACK NUMBER
   -18441 B7F7         LAST USED SLOT
   -18440 B7F8         LAST USED DRIVE
          2D           CURRENT SECTOR
          2E           CURRENT TRACK

THE STARTING ADDRESS OF A BINARY FILE
PRINT "ADDRESS = "; PEEK (43634) + PEEK (43635) * 256

THE LENGTH OF A BINARY FILE
PRINT "LENGTH = "; PEEK (43616) + PEEK (43617) * 256

TO EXEC THE CATALOG ROUTINE FROM THE MONITOR TYPE A56EG
```

```
================================================================================
DOCUMENT dosless.txt
================================================================================

1

---------------------------------------
[Ctrl-S pauses/Space=quit]

1
0

   WHENEVER YOU INITIALIZE A BLANK DISK, A COPY OF DOS IS WRITTEN ONTO IT.
THIS ENSURES THAT THE DISK IS BOOTABLE.  BUT SOME DISKS ARE NEVER BOOTED.
THEY'RE USED ONLY TO STORE PROGRAMS, TEXT FILES OR OTHER DATA.  IF DOS COULD
BE ELIMINATED FROM THESE DISKS, YOU'D GAIN EXTRA STORAGE SPACE FOR OTHER FILES.

   SUCH A DOS-LESS DISK CAN BE CREATED VERY EASILY.  THE METHOD DESCRIBED
BELOW WAS POSTED ON COMPUSERVE BY BILL STEINBERG, A MEMBER OF THE APPLE
INTEREST GROUP OVER THERE.  THE PROCEDURE WORKS WITH ANY 48K APPLE II WITH
DOS 3.3.

   FIRST, BOOT ANY STANDARD APPLE DOS DISK IN ORDER TO LOAD DOS INTO MEMORY.
IF A HELLO PROGRAM RAN, EXIT IT AND GET INTO APPLESOFT.  NOW TYPE IN THE
FOLLOWING SIX POKES:

POKE -20734, 234
POKE -20733, 234
POKE -20732, 234
  (THOSE POKES PREVENT DOS FROM BEING WRITTEN TO THE DISK DURING THE INIT
PROCESS.)

POKE -20813, 4
  (CLEARS THE VTOC SECTOR BIT MAP DOWN TO TRACK 1 INSTEAD OF TRACK 3.)

POKE -23188, 208
POKE -23187, 3
  (EXITS THE INIT ROUTINE WITHOUT SAVING A HELLO PROGRAM.)

   THAT'S ALL.  NOW INITIALIZE A BLANK DISK USING THE INIT COMMAND.  (DON'T
TRY THIS ON A DISK WHICH ALREADY CONTAINS DATA!)  THE NEWLY INITIALIZED DISK
WILL NOT BE ABLE TO BOOT, BUT IT WILL PROVIDE 2 TRACKS (32 SECTORS) OF EXTRA
SPACE FOR PROGRAMS OR OTHER FILES.  YOU MAY WISH TO LABEL SUCH DISKS AS
"DATA DISKS" OR "NON-BOOTABLE" OR SOMETHING SIMILAR.  TO USE THESE DISKS
IN THE FUTURE, YOU MERELY LOAD DOS INTO YOUR APPLE WITH A REGULAR, BOOTABLE
DISK, THEN INSERT ONE OF YOUR DOS-LESS DISKS TO LOAD OR SAVE FILES ON IT
AS YOU WOULD WITH ANY OTHER DISK.

---------------------------------------

Enter (1-10, M=Menu, Q=Quit) :
```

```
===============================================================================
DOCUMENT emu.pt.update
===============================================================================

 Brd ->IIGS Technical Sub
Numb ->50 of 50
 Sub ->** read **
  To ->All
From ->The Martyr (#11)
Date ->12/24/87  07:58:07 PM

Hello, with the release of ProTERM v1.9p (only a few steps from 2.0) there have
been a couple of noted problems and concerns.

Number 1; It still does not have an USRobotics HST driver in the configuration
        program.  Well this problem has already been solved.  Myers (of The
        9600 Club) re-wrote the drivers to work correctly.  I just put up the
        Super Serial Card one, the Modem Port one is done but I have yet to
        upload it.  They both work.  Remember the patch for v1.2 will not work
        with v1.9p.  Don't bother trying it, their formats are different.

Number 2; The editor DOES work.  It can save text/appleworks format (and it
        works fine)  It is extremely radical.  I like it better than Appleworks
        The editor is version 2.0, the program is v1.9p. (like I said, only
        a few steps away)

Number 3; Greg has fixed many upsetting things in proterm.  For example,
        when dialing after a few seconds if your modem doesn't connect
        for whatever reason it will keep that boards' dial screen there.
        (instead of returning you to the board list)

Number 4; It supports Y-modem CRC when downloading.  This will not speed
        anything up, but will have better error checking.  Transit finally
        works.  It fairly fast too!

Number 5; ** THE MOST IMPORTANT ** ProTERM supports mousetext and a sound
        generator, this is a terminal emulation.  Like VT-100 and Datamedia
        are.  This is UNREAL!!

        What is mousetext?  Remember when you got your //e enhanced about
        a year ago?  And Appleworks could show like open and closed Apples at
        the main menu!  This will revolutionize bulletin boards.

        ONLY WITH PROTERM V1.9 - V2.0
        -----------------------------

        The Sound Generator --> Supports pitch, tempo, tone. It is feasible
        to actually write a song that could be sent over the phone to the
        person on the other end.

        ITS TERMINAL EMULATION:  ProTERM Special, so configure your system to
        that.

        Compatibility: Semi-compatible with Datamedia 1500.  It uses
                        the same hi and lo lite control characters (inverse)
                        left, right, down are the same.  The rest are all
                        different. (except for control-g of course)
```

```
(*note: the up control character is ^K (just like the arrows))

        The Future: You'll be using your mouse pretty soon if Greg keeps
                this up.  Learn to use these radical terminal emulations,
                you'll find boards can and will be MUCH better.

        The First: The first board in the world supports mousetext/ProTERM
                special emulation.

                                Brave New World

                                 BBS/Y-xmodem
                                  12/24/9600
                                     15meg
                [Apple*link - Datamedia 1500, ProTERM Special]

                              [ProTERM Emulator]

                                 617-849-0644
                                    (come and see mousetext in action.)

Thank you very much, this is very important.  Would you like to use mousetext
and the sound generator in your system?  I will release a text file soon with
all the compatible control characters.

For making this information possible (and alerting me to the fact that
ProTERM special emulation is powerful):  Ralph Kramden

And the person on line while I figured it out: Sound Wave, thanks for the
moral support. (haha)

thanks,

The Martyr of The 9600 Club
```

==
DOCUMENT errors.app
==

A COMMENT ON ERROR TRAPS
BY NICK FOTHERINGHAM
FROM THE APPLE BARREL, JULY'82
I.A.C.-TC

YOU HAVE FINALLY GOTTEN ALL OF THE BUGS OUT OF THAT SPECIAL PROGRAM THAT HAS KEPT YOU IN SECLUSION FOR THE PAST SEVERAL WEEKS. IT DOES EXACTLY WHAT YOU WANT IT TO DO, AND YOU ARE READY TO IMPRESS SOMEONE WITH IT. YOU BEG YOUR BOSS TO TAKE TIME FROM HIS BUSY SCHEDULE FOR A SESSION WITH YOUR APPLE, AND AFTER TEN MINUTES OF ROUTINE DATA ENTRY, YOUR PROGRAM IS NEARING ITS FLASHY FINALE. THE NEXT QUESTION APPEARS: "HOW MANY SIDES ON AN OCTOGON?" AS YOUR BOSS ENTERS "E..I..G...", YOU STIFLE, "NOT THAT KEY, YOU DUMMY, THE '8'". TOO LATE... THE APPLE HAS ALREADY RESPONDED WITH A "TYPE MISMATCH" MESSAGE AND SHUT YOUR PROGRAM.

ONE PURPOSE OF AN "ERROR TRAP" OR "ERROR HANDLING

ROUTINE" IS TO HELP PREVENT SUCH EMBARRASSING SITUATIONS. YOUR APPLE'S BASIC INTERPRETER ALREADY HAS SEVERAL BUILT-IN ERROR TRAPS WHICH WERE DESIGNED TO PROTECT THE SYSTEM FROM YOUR UNREASONABLE REQUESTS, SUCH AS ATTEMPTS TO DIVIDE BY ZERO OR TO EXCEED TO SYSTEM'S CAPACITY ("STRING TOO LONG", "OVERFLOW", "FORMULA TOO COMPLEX", "OUT OF MEMORY"). FORTUNATELY FOR MANY APPLICATIONS, THESE TRAPS CAN BE AVOIDED BY USING THE ONERR GOTO.....POKE 216,0 COMMANDS. ONERR GOTO... DISABLES THE SYSTEM'S INTERNAL ERROR HANDLING ROUTINE AND, UPON ENCOUNTERING AN ERROR, TRANSFERS PROGRAM PROCESSING TO A STATEMENT DEFINED BY THE GOTO STATEMENT, TYPICALLY A REPLACEMENT ERROR HANDLING ROUTINE OF YOUR DESIGN. THE POKE 216,0 COMMAND REINSTATES THE SYSTEM'S ERROR HANDLING ROUTINE.

FOR MANY BEGINNING PROGRAMMERS, DISABLING THE SYSTEM'S ERROR HANDLING ROUTINE, ONLY TO REPLACE IT WITH ONE THAT YOU MUST DESIGN AND WHICH USES SOME OF YOUR PRECIOUS RAM MEMORY SEEMS LIKE LUNACY. THE MAJOR REASON FOR DOING SO IS THAT MOST OF THE ERRORS TO WHICH THE SYSTEM REACTS NEED NOT BE FATAL TO

YOUR RUN. THE COMPUTER VIEWS THESE ERRORS AS FATAL BECAUSE THE CONTEXTS IN WHICH THEY MAY OCCUR ARE SO DIVERSE THAT THE ONLY GENERAL SOLUTION THAT ENSURES PROTECTION TO YOUR COMPUTER IS TO TERMINATE YOUR RUN. HOWEVER, WITHIN YOUR PROGRAM THE CONTEXT WITHIN WHICH AN ERROR MAY OCCUR CAN OFTEN BE MUCH MORE NARROWLY DEFINED, AND NONFATAL SOLUTIONS MAY BE DEVELOPED. SOME OF THESE SOLUTIONS ARE DESCRIBED BELOW.

ONE OF THE MOST COMMON APPLICATIONS FOR ERROR TRAPS IS TO GUARD YOUR PROGRAM AGAINST TYPING ERRORS DURING DATA ENTRY FROM THE KEYBOARD. MOST SUCH ERRORS CAN BE RESOLVED WITHOUT ABORTING YOUR PROGRAM BY DESIGNING THE PROGRAM TO RECEIVE ALL INPUT AS A STRING VARIABLE, SAY A$. BECAUSE A$ WILL ACCEPT INPUT FROM NEARLY EVERY KEY (EXCEPT RESET) WITHOUT A TYPE MISMATCH ERROR, IT IS PREFERABLE TO A OR A% AS AN INPUT VARIABLE. YOU MAY THEN TEST THE INPUT TO SEE IF A RETURN HAS BEEN ENTERED (A$=""), TO SEE IF A NUMBER HAS ABEEN ENTERED (ASC(A$)>47 AND ASC(A$)<58. IF THE DESIRED NUMERICAL INPUT HAS BEEN ENTERED, YOU MAY THEN CONVERT THE INPUT TO ITS

NUMERICAL EQUIVALENT (A=VAL(A$) OR A%=INT(VAL(A$))) AND THEN TEST TO SEE IF THIS VALUE IS WITHIN THE RANGE THAT YOU EXPECTED AS AN ANSWER TO YOUR QUESTION

(A%>0 AND A%<5).

ONE OF THE GREAT ADVANTAGES OF OWNING YOUR OWN COMPUTER SYSTEM ON WHICH YOU RUN PROGRAMS INTERACTIVELY IS THAT YOU CAN USUALLY TRAIN THE SYSTEM TO COME BACK TO YOU FOR HELP WHEN IT HAS A COMPLAINT INSTEAD OF JUST DYING.   WHEN A "FATAL" PROBLEM IS ENCOUNTERED, SUCH AS AN ATTEMPT TO DIVIDE BY ZERO, AN ERROR TRAP CAN BE USED TO PRINT AN ERROR MESSAGE OF YOUR CHOOSING AND THEN GIVE YOU AN OPPORTUNITY TO CHANGE THE DENOMINATOR TO A NON-ZERO NUMBER AND CONTINUE THE CALCULATION OR TO ABORT THAT PROGRAM SEGMENT (E.G. RETURN TO THE MENU).

GOOD PROGRAMS SHOULD NEVER "CRASH". EVEN WHEN THEY FAIL TO COMPLETE THE TASK FOR WHICH THEY WERE DESIGNED, THEY SHOULD REACH A CONTROLLED ENDING WHICH PROVIDES A DETAILED DESCRIPTION OF WHAT WENT WRONG AND AN OPPORTUNITY TO FIX IT BEFORE ENDING.   SINCE MOST OF US WRITE PROGRAMS WITH THE

EXPECTATION THAT OTHERS WILL RUN THEM, WE SHOULD GET IN THE HABIT OF USING ERROR TRAPS ROUTINELY, AND WE SHOULD INSIST ON SUCH PROGRAMMING STYLE IN THE COMMERCIAL SOFTWARE WE BUY.

```
==============================================================================
DOCUMENT errors.txt
==============================================================================

2

---------------------------------------
[Ctrl-S pauses/Space=quit]

2
0
```

A COMMENT ON ERROR TRAPS
BY NICK FOTHERINGHAM
FROM THE APPLE BARREL, JULY'82
I.A.C.-TC

YOU HAVE FINALLY GOTTEN ALL OF THE BUGS OUT OF THAT SPECIAL PROGRAM THAT HAS KEPT YOU IN SECLUSION FOR THE PAST SEVERAL WEEKS. IT DOES EXACTLY WHAT YOU WANT IT TO DO, AND YOU ARE READY TO IMPRESS SOMEONE WITH IT. YOU BEG YOUR BOSS TO TAKE TIME FROM HIS BUSY SCHEDULE FOR A SESSION WITH YOUR APPLE, AND AFTER TEN MINUTES OF ROUTINE DATA ENTRY, YOUR PROGRAM IS NEARING ITS FLASHY FINALE. THE NEXT QUESTION APPEARS: "HOW MANY SIDES ON AN OCTOGON?" AS YOUR BOSS ENTERS "E..I..G...", YOU STIFLE, "NOT THAT KEY, YOU DUMMY, THE '8'". TOO LATE... THE APPLE HAS ALREADY RESPONDED WITH A "TYPE MISMATCH" MESSAGE AND SHUT YOUR PROGRAM.

ONE PURPOSE OF AN "ERROR TRAP" OR "ERROR HANDLING

ROUTINE" IS TO HELP PREVENT SUCH EMBARRASSING SITUATIONS. YOUR APPLE'S BASIC INTERPRETER ALREADY HAS SEVERAL BUILT-IN ERROR TRAPS WHICH WERE DESIGNED TO PROTECT THE SYSTEM FROM YOUR UNREASONABLE REQUESTS, SUCH AS ATTEMPTS TO DIVIDE BY ZERO OR TO EXCEED TO SYSTEM'S CAPACITY ("STRING TOO LONG", "OVERFLOW", "FORMULA TOO COMPLEX", "OUT OF MEMORY"). FORTUNATELY FOR MANY APPLICATIONS, THESE TRAPS CAN BE AVOIDED BY USING THE ONERR GOTO.....POKE 216,0 COMMANDS. ONERR GOTO... DISABLES THE SYSTEM'S INTERNAL ERROR HANDLING ROUTINE AND, UPON ENCOUNTERING AN ERROR, TRANSFERS PROGRAM PROCESSING TO A STATEMENT DEFINED BY THE GOTO STATEMENT, TYPICALLY A REPLACEMENT ERROR HANDLING ROUTINE OF YOUR DESIGN. THE POKE 216,0 COMMAND REINSTATES THE SYSTEM'S ERROR HANDLING ROUTINE.

FOR MANY BEGINNING PROGRAMMERS, DISABLING THE SYSTEM'S ERROR HANDLING ROUTINE, ONLY TO REPLACE IT WITH ONE THAT YOU MUST DESIGN AND WHICH USES SOME OF YOUR PRECIOUS RAM MEMORY SEEMS LIKE LUNACY. THE MAJOR REASON FOR DOING SO IS THAT MOST OF THE ERRORS TO WHICH THE SYSTEM REACTS NEED NOT BE FATAL TO

YOUR RUN. THE COMPUTER VIEWS THESE ERRORS AS FATAL BECAUSE THE CONTEXTS IN WHICH THEY MAY OCCUR ARE SO DIVERSE THAT THE ONLY GENERAL SOLUTION THAT ENSURES PROTECTION TO YOUR COMPUTER IS TO TERMINATE YOUR RUN. HOWEVER, WITHIN YOUR PROGRAM THE CONTEXT WITHIN WHICH AN ERROR MAY OCCUR CAN OFTEN BE MUCH MORE NARROWLY DEFINED, AND NONFATAL SOLUTIONS MAY BE DEVELOPED.

SOME OF THESE SOLUTIONS ARE DESCRIBED BELOW.

ONE OF THE MOST COMMON APPLICATIONS FOR ERROR TRAPS IS TO GUARD YOUR PROGRAM AGAINST TYPING ERRORS DURING DATA ENTRY FROM THE KEYBOARD. MOST SUCH ERRORS CAN BE RESOLVED WITHOUT ABORTING YOUR PROGRAM BY DESIGNING THE PROGRAM TO RECEIVE ALL INPUT AS A STRING VARIABLE, SAY A$. BECAUSE A$ WILL ACCEPT INPUT FROM NEARLY EVERY KEY (EXCEPT RESET) WITHOUT A TYPE MISMATCH ERROR, IT IS PREFERABLE TO A OR A% AS AN INPUT VARIABLE. YOU MAY THEN TEST THE INPUT TO SEE IF A RETURN HAS BEEN ENTERED (A$=""), TO SEE IF A NUMBER HAS ABEEN ENTERED (ASC(A$)>47 AND ASC(A$)<58. IF THE DESIRED NUMERICAL INPUT HAS BEEN ENTERED, YOU MAY THEN CONVERT THE INPUT TO ITS

NUMERICAL EQUIVALENT (A=VAL(A$) OR A%=INT(VAL(A$))) AND THEN TEST TO SEE IF THIS VALUE IS WITHIN THE RANGE THAT YOU EXPECTED AS AN ANSWER TO YOUR QUESTION (A%>0 AND A%<5).

ONE OF THE GREAT ADVANTAGES OF OWNING YOUR OWN COMPUTER SYSTEM ON WHICH YOU RUN PROGRAMS INTERACTIVELY IS THAT YOU CAN USUALLY TRAIN THE SYSTEM TO COME BACK TO YOU FOR HELP WHEN IT HAS A COMPLAINT INSTEAD OF JUST DYING. WHEN A "FATAL" PROBLEM IS ENCOUNTERED, SUCH AS AN ATTEMPT TO DIVIDE BY ZERO, AN ERROR TRAP CAN BE USED TO PRINT AN ERROR MESSAGE OF YOUR CHOOSING AND THEN GIVE YOU AN OPPORTUNITY TO CHANGE THE DENOMINATOR TO A NON-ZERO NUMBER AND CONTINUE THE CALCULATION OR TO ABORT THAT PROGRAM SEGMENT (E.G. RETURN TO THE MENU).

GOOD PROGRAMS SHOULD NEVER "CRASH". EVEN WHEN THEY FAIL TO COMPLETE THE TASK FOR WHICH THEY WERE DESIGNED, THEY SHOULD REACH A CONTROLLED ENDING WHICH PROVIDES A DETAILED DESCRIPTION OF WHAT WENT WRONG AND AN OPPORTUNITY TO FIX IT BEFORE ENDING. SINCE MOST OF US WRITE PROGRAMS WITH THE

EXPECTATION THAT OTHERS WILL RUN THEM, WE SHOULD GET IN THE HABIT OF USING ERROR TRAPS ROUTINELY, AND WE SHOULD INSIST ON SUCH PROGRAMMING STYLE IN THE COMMERCIAL SOFTWARE WE BUY.

Enter (1-10, M=Menu, Q=Quit) :

```
==============================================================================
DOCUMENT expandca.app
==============================================================================

=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%
                    Expanding your Apple Cat //
                              By:
                     ((%>> The Ware-Wolf <<%))
             (Hi-Res<>Hijackers/The 202 Alliance/WareBusters!)
%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%
Apple Manor___<716>/654-POOF! (10 Meg) -- The Outpost___<312>/441-6957 (10 Meg)
%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%=%
```

The Apple Cat // modem is by far the most expandable modem on the market today. Of course it's also the choice modem of pirates because of it's inexpensive half-duplex 1200 baud capabilities. The expansion module available for the cat has several very useful functions. Rather than shelling out $30 bucks for one which you may only use a few of the features this file tells you how to build just certain features or even the whole package.

First off you'll need some basic knowledge and tools. As for the knowledge you'll need to know how to solder pretty well, you'll also proabibly have to know DC from Hz and +12V from RS232. Ok now, If you can handle that that, you'll need these tools:

- A soldering iron and solder
- A flat, 14 wire, female cable. Preferably multi-colored.
    * Note: Single strands of wire will do but they risk damaging your cat.

We'll be connecting the wires to the J2 connector (see owner's manual, fig. 2). Remember that there are 25 pins on this connector. Each pin numbered starting with pin 1 in the rear of your computer and pin 25 closest to the keyboard. We'll only be working with the first 14 pins. The rest are for the 212 and speech synthesizer cards.

Here is a table which tells something about each pin:

Pin #	Function	Direction	Feature
01	Transmit Data	Output	EIA-RS232C Printer interface
02	Receive Data	Input	
03	Clear to Send Signal	Input	
04	Signal Ground	GND	
05	AC line reference (60Hz)	Input	BSR Remote control
06	Signal Ground	GND	
08	+12V DC	Output	
09	120 KHz Control Signal	Output	
07	+12V DC	Output	Off-Hook LED
12	LED Drive	Output	
10	Tape Recorder Control	Input	Tape Recorder
11	Tape Recorder Control	Output	
12	Audio Signal to Tape	Output	
14	Signal Ground	GND	

* Note: This table corrects several errors which occur in the table in the Owner's Manual.

Bulidin` the On/Off hook indicator
==================================
Required parts: 12V DC LED
==================================
This is the most inexpensive and simple of the projects. All you must do is connect the wire leading from pin 7 to the positive pole of the LED and connect pin 12 to the remaining pole. Solder connections firmly and whenever the modem is off-hook the LED will light.

Hooking up a tape player
========================
Required parts: Tape Recorder with adjustable record level, 3.5 mm patch cable; male on one end; stripped on the other, Patch cable with 2.5 mm plug on one end;stripped on the other.
========================
This is proabibly the most useful feature. With this feature you may listen in on your cat. Such as when calling a board you'll never have to pick up the phone. You also might want to do an answering machine. I'll tell you more about that later.

To build this you must take the wires leading from pins 10 & 11 and connect them to the stripped ends of your 2.5 mm patch cable. Now take the wires leading from pins 13 & 14 and connect them to the stripped ends of your 3.5 mm patch cable. ** Note: You may have to reverse which pin goes to which wire on each cable if it doesn't work at first. Now, simply plug the 3.5 mm plug into the Mic jack on the tape recorder and plug the 2.5 mm plug into the Rem jack on the tape recorder.

To use this you just press the Rec button(s) on your tape recorder. On most tape recorder you'll be able to hear what is going on when the modem picks up the phone. You'll notice that the tape does not move when you press record, you must do a POKE 49313,31 (Default = 0) to turn on the tape. That is how you make your answering machine. ** Note: I have included an answering machine program at the end of his file.

Bulidin` the EIA-RS232C printer interface
===
Required parts: Serial printer, RS232C cable
===
This is pretty difficult to explain. We'll start by looking at the RS232C port on the back of your printer. This port has two rows of holes. One row has 12 holes and the other has 13. We'll number these holes by going left to right the first holes are 1 to 13 on the largest row, next go to the left of the smaller row and number from 14 to 25. Not all of these holes will be used. This chart tells which wire goes to which hole:

Pin #	Hole(s)
01	12
02	11
03	19+3 (19 first)
04	07

Hooking up the BSR Remote Transformer
=====================================
Required Parts: BSR Remote Transformer
=====================================
      ** Note: This is really quite dangerous and I recommend if you wish to use this function and are unsure of your abilities that you buy an expansion module.

            Now, look at the square end of your transformer. Each hole should have a number next to it. If you don't see these numbers than just number counter-clockwise starting at the bottom left corner (notch facing the floor). There is really no good way to get the wires to stay in these holes. You may want to go to Radio Shack and look for something. Anyways be sure the transformer is not plugged into the wall and connect each pin to each hole as shown:

Pin #5--> Hole #3
Pin #6--> Hole #1
Pin #8--> Hole #2
Pin #9--> Hole #4

**Caution: Be sure that no wire touches another wire!

      To use this you must have at least one of those modules which come with the real BSR Command things. There is a program on your Com-Ware disk to control this.

**Caution: When working on these features be sure to connect them to the pins last or else damage to you or your cat may occur.

Here is the answering machine program I mentioned earlier:

```
10  REM  -> A WARE-WOLF PRODUCTION
20  POKE 49314,0: POKE 49313,0
40 S = 38142:P = 38141:M = 33056:T = 33055:C = 22357:A = 38131:D$ =  CHR$ (13) +
70 KB =  - 16384:PR =  - 16211:CC = 49168
80  HOME : PRINT CA
90  IF      PEEK (KB) = 195 THEN ZZ =  PEEK (CC): RUN
110  IF  PEEK (KB) = 212 THEN ZZ =  PEEK (CC): GOTO 160
120  IF  PEEK (KB) = 209 THEN  PRINT  CHR$ (8): POKE 49168,0: END
130  IF  PEEK (PR) / 2 =  INT ( PEEK (PR) / 2) THEN 90
140  PRINT "Sam:";: INVERSE : PRINT "Receiving Call": NORMAL
160  POKE 49314,2: FOR X = 1 TO 3500: NEXT
170 SA$ = "HELLO.THERE.YOU HAVE.REACHED.THE.WARE.WOLFS.COMPUTER": GOSUB 400: CAL
180 SA$ = "NOW.LISTEN UP.SUNNY.IF.YOU DON'T.LISTEN.WE.MIGHT.HAVE TO.KICK YOUR AS
190 SA$ = "I.WON'T.HANG.UP.TILL.YOU.ARE FINISHED.LEAVING.YOUR.MESSAGE": GOSUB 32
200 SA$ = "REMEMBER.TO.WAIT.FOR.THE.BEEP.": GOSUB 380: CALL A
210 SA$ = "BYE": GOSUB 300: CALL A: GOSUB 320: CALL A: GOSUB 340: CALL A: GOSUB
220  FOR Z = 1 TO 190:V = ( PEEK ( - 16224) - 15): IF ((V / 16) / 2) <  >  INT (
230  PRINT Z: IF Z =  > 190 THEN 250
240  GOTO 220
250 SA$ = "THANKS FOR THE MESSAGE": CALL A
260  POKE 49314,0: POKE 49313,0
270 CA = CA + 1
280  GOTO 40
300  REM     ***ELF***
```

```
310  POKE T,110: POKE M,160: CALL C: POKE S,72: POKE P,64: RETURN
320  REM    ***ROBOT***
330  POKE T,190: POKE M,190: CALL C: POKE S,92: POKE P,60: RETURN
340  REM      ***STUFFY GUY***
350  POKE T,110: POKE M,105: CALL C: POKE S,82: POKE P,72: RETURN
360  REM       ***OLD LADY***
370  POKE T,145: POKE M,145: CALL C: POKE S,82: POKE P,32: RETURN
380  REM     ***E.T.***
390  POKE T,150: POKE M,200: CALL C: POKE S,100: POKE P,64: RETURN
400  REM    ***REGULAR***
410  POKE T,128: POKE M,128: CALL C: POKE S,74: POKE P,64: RETURN
```

To use this program first, EXEC it into basic and save it. Next boot up Sam Knobs and select the text input version. Now when run this program will put a 0 in the upper-left corner of the screen. This is how many calls you have had so far. To test the program just hit "T" to clear the call count hit "C" to quit hit "Q". It after the little greeting message it waits until there is no sound for about 6-7 seconds. So people can leave messages of unlimited length. I included the pokes for different voices so you can be creative with your messages.

==========
The End...
==========

```
==============================================================================
DOCUMENT futrae.app
==============================================================================
```

"The best ideas are the ideas that help people."
 PHido PHreaks PResent...
 The Future Evolution of Ascii Express
 By the Silver Ghost

August, 1987: Version 5.0

Offers ARC storage, which will compress or decompress files in IBM ARC format--handy for text files! Included is XYZMODEM32, a hot new protocol that allows bi-directional file transfer and chatting at the same time.

January, 1988:   Version 5.2Q

Automatically ARCs and de-ARCs everything on the disk, for an average savings of 40%. Gives individ ual-password access to specific files, for private mail-sending. The new transfer protocol is WCXYQMODEM64, which supports conference and three-way calls, so you can send the same file to more than one person at the same time.

March, 1988: Version 5.2S

Improved ARCing--average savings is now up to 50%.

November, 1988: Version 6.02

The ARC is now compatible with all earlier versions, so you can use all your old disks. Now runs AEDOS instead of ProDOS; AEDOS will read or write to ProDOS, Pascal, CP/ M, or DOS 3.3 disks automatically.   The new protocol is YXmodem-7, a software patch allowing 1200 baud modems to run at 2400 baud, while sending up to eight files bidirectionally simultaneously, while chatting. Up to 256 people can conference and receive these files. A LOG feature allows the AE sysop to print out a complete log of every caller's activities.

July, 1989: Version 6.57

ARC now crunches files to 30% of their original size using a special fractal procedure. All earlier ARCs are of course supported. In answer code, AE traces all incoming calls, and puts the caller's phone number into the LOG. For their convenience, Apple-Cat owners may use the PHREAK feature, which sends a guaranteed-to-be-untraceable-or-your-money-back 2600 Hz tone, all owing free calls.   AT&T files suit and wins; it's appealed. Meanwhile AEPro6.57 is selling like hotcakes. X-Marks-The-SpotModem will not only support conference calling, it will, quasi-legally, create conferences for you to use (Apple-Cat owners only).

December, 1989: Version 7.00

The appeal of the AT&T suit is cancelled, as Southwestern Data Systems buys out AT&T in an unfriendly takeover. Version 7.00 has a multi-purpose PHREAK command that's compatible with all Hayes-compatible modems. X-RatedModem8 will nybble-transfer the entire contents of anything that fits on your disk drive, from an off-the-shelf copy of Flight Simulator VI to your socks. The special software-generated tones are above the range of human hearing--so while you send up to thirty-two files or disks multidirectionally between up to 4096 people at 9600 baud, you can pick up the phone and chat voice with everyone else. ARC now compresses data down to 10% of its original size.

August, 1990: Version 8.00

Assuming that both parties have a Write-Once-Read-Many Compact Disk Drive, XXXModem allows sending of any CD, from a spreadsheet to Pole Position III to the Beatles, across the phone lines. Capitol Records files suit and is bought up in a corporate takeover. On-line hacking help is available for any of over a hundred types of mainframes.   ARC squeezes data to 5% of its original size, allowing 40 megs to fit on one of the new 3 1/2" disks. The LOG trace feature now offers a comprehensive Federal Agency Search, which will identify FBI offices by phone number and, if desired, reduce their access accordingly. Version 8.00 supports up to four modems connected simultaneously to the Apple //SX, with no significant slowing of speed.

April, 1991: Version 9.05

The LOG trace now includes the home phone numbers of 95% of all employees of the government. The U.S. Government sues SDS and is bought up in an unfriendly takeover; all laws regarding the telephone system are repealed. With a simple video camera hookup, Special-F-XModem will use the Super-Super-Hi-Res screen (4096x1024, 64 million colors) to display the person on the other end of the line while up to eight modems transfer up to 256 files or disks each simultaneously, multidirectionally, between 65,536 people at 96,800 baud. The telephone is now obsolete--the internal speakers allow listening in, while the internal microphones pick up the user's voice, with Dolby Z filtering automatically applied to eliminate line noise. The latest automatic ARC scrunches files to an av erage 3% of their original size--about 70 megs fit on the 2.5" disks.

This fantasy courtesy of Thieves' World FIDO, 616-344-7218, blah blah blah.
Just call the number, okay? It won't kill you.   EOF!

==
DOCUMENT icon.convert
==

Converting Apple IIGS Icons to Clip Art
by Marty Knight

Materials needed:
   Icons - available from AGR or AUT
   C1 Pic Saver CDA - available from APR
   DIcEd Desktop Icon Editor v1.2 - available from AUT
   SHRConvert v2.1 (to change filetype) - available from AGR
   Paint program that will handle 640 mode graphics

Procedure:
1.   Collect the necessary materials.

2.   Install C1 Pic Saver CDA by copying it to your */SYSTEM/DESK.ACCS folder and rebooting. It will appear on the CDA menu as SHR C1 Saver. This CDA will take a "snapshot" of the SHR screen and save it to your disk. The name of the file will be something like Screen.x where x is a number. Each file you save will be 65 blocks long, so be sure your disk has enough room. You can save at most 20 pictures with this CDA. If you need to save more, you must rename the Screen.x files and reboot. Do not use this CDA if the SHR screen is not visible.

3.   Launch DIcEd Desktop Icon Editor.

4.   Open your first icon file. It should look something like Figure 1.

5.   Resize the window so it is as small as possible while still leaving the icon fully visible. Then move the window to the upper left corner of the screen so that it looks like Figure 2.

6.   Open another icon file, resize its window, and move it up next to the first window. Place the windows close to each other, but not so close that the icons overlap. Continue doing this until the screen is filled. You should be able to fit about 12 icons on one screen as shown in Figure 3.

7.   Access the CDA menu (OA-control-Esc) and select the SHR C1 Saver CDA. Your disk will spin as a snapshot of the DIcEd screen is saved to disk. The first picture saved will be called Screen.0, the second picture saved will be called Screen.1, and so on up to Screen.9. After ten pictures are saved the pictures will be saved using the names Screen.10, Screen.20, and so on up to Screen.90.

8.   Close all the DIcEd windows (press OA-K for each one), and repeat the process until you run out of icons or space on the disk.

9.   Now launch SHR Convert. Select "Change file attributes" from the File Menu.

10.   Select the first screen saved (Screen.0) with the CDA. When you select it, you will see a screen like the one shown in Figure 4. Notice that the New filetype is highlighted and says BIN. This is what you will change.

11.   Type "$C1" into the New filetype field. When you are done, it will look like Figure 5. Press Return.

12.   Repeat the process for each of your saved screens.

13. Now you can launch your favorite paint program and edit the screens. You should delete all parts of the screen except the icon graphics. Then rearrange the graphics to pack as many as you can onto one screen. You will probably want to copy and paste icon graphics from some of the other screens. Remember to save each picture back to disk when you are done. I find HyperStudio an excellent program for this kind of work. I can edit the screens individually and group the graphics together. Then I can add whole groups of icon graphics as clip art.

14. You're finished. You now have a lot of small graphics that make excellent buttons for HyperStudio. They are also great for use as clip art on page layouts.

```
================================================================================
DOCUMENT iigsprob.hum
================================================================================
```

An Apple //GS Annoyance

Here is a little dismayig problem that has virtually made me loose all respect n Apple all together.

I just recently purchased my Apple //gs system with the works, including that nice $10 muffin fan Apple sells for $50. After playing around with my //GS for a day or two, I decided to plug a pair of headphones into the headphone jack to hear what it sounds like. As soon as I plugged the headphones in I noticed this terribly loud electrical hum coming through the headhones, I pay much attention to this, thinking that it was the way it was supposed to be.

Last week, I bought the MD-Ideas Supersonic stereo card and promptly hooked it up to my stereo system. Here's where the problems start. I thought that the stereo card would solve the hum problem, but no such luck, the hum was even louder and more annoying as ever. I promptly called up MD-Ideas, and they told me that the problem was caused by the //gs fan, that is why their fan sits on the outside of the //gs and runs on its own power supply. Then, I proceeded to call up my Apple Dealer (B.C. Communications in Huntington L.I.) and the owner told me that, "Oh, I have encountered this problem before, that is why Apple doesn't make an amplifier for the //gs".

Well, I hardly took this explaination seriously and called up Apple Customer Relations in C.A., they put me in touch with their technical support line, and I got an answer...Here it is: "We can't do anything about it, they all have this problem, we suggest that you unhook the fan and leave the cover off the machine". I told them that unhooking the fan would void my warranty since I have more than 3 peripheral cards plugged in, they responded with an "We know it voids the warranty but nothing can be done."

It can hardly be beleived that Apple would put out an overpriced product that works halfway and not be willing to fix it or at least offer our money back. If we use it with the fan hooked up, we can't fully enjoy our machines, if we use it without the fan, we void the warranty and risk having to pay Apple's overpriced repair bills. Please protest the bit of shamefull behavior and DON'T buy the //GS fan from Apple, and if you allready have, demand a fix or your money back!

After all, the "S" in "Apple IIGS" stands for sound doesn't it? Or maybe to Apple the "S" stands for swindle...

PLEASE IF YOU ARE A MEMBER OF ANY ONLINE BOARD WHERE THERE ARE APPLE USERS (ESPECIALLY COMPUSERVE, THE SOURCE AND GENIE), UPLOAD THIS FILE.PPLE USERS (ESPECIALLY COMPUSERVE, THE SOURCE AND GENIE), UPLOAD THIS

(>

```
==============================================================================
DOCUMENT index.html
==============================================================================

<HTML>
<TITLE>T E X T F I L E S</TITLE>
<BODY BGCOLOR="#000000" TEXT="#00FF00" LINK="#00FF00" ALINK="#00FF00"
VLINK="#00FF00">
<H1>Apple II Textfiles</H1>
With the introduction of the Apple II family of computers, the wonders of
programming, communicating, and just plain geeking out became affordable for
an entire generation of budding enthusiasts and their families. By the end of
the 70's an entire culture had risen up around the Apple II, and the energy
of thousands of hardware and software hackers went into learning every last
op-code and settable switch within the machine.
<P>
It can't be discounted that Apple's successful foray into the educational
market resulted in schools countrywide brimming with Apple IIs, and social
groups collecting around the labs after school hours. All manner of things
happened there, some documented below.
<P>
These files range from explicit memory maps of the Apple II to long tutorials
on how to "crack" games, that is, remove all copy protection and make the game
easier to distribute between other pirates.
<P>
<TABLE WIDTH=100%>
<TD BGCOLOR=#00FF00><FONT COLOR=#000000><B>Filename</B><BR></FONT>
<TD BGCOLOR=#00DD00><FONT COLOR=#000000><B>Size</B><BR></FONT>
<TD BGCOLOR=#00AA00><FONT COLOR=#000000><B>Description of the
Textfile</B><BR></TR>

<tab indent=60 id=T><br>
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="DOCUMENTATION">DOCUMENTATION</A><TAB
TO=T><TD> DIRECTORY &nbsp;<BR><TD> "Soft Dox" for Apple Programs
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="GENIELAMP">GENIELAMP</A><TAB TO=T><TD>
DIRECTORY &nbsp;<BR><TD> Archive of the Genielamp A2, the GEnie Apple II
Roundtable
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="WALKTHROUGHS">WALKTHROUGHS</A><TAB TO=T><TD>
DIRECTORY &nbsp;<BR><TD> Walkthroughs of Apple II Specific Adventures
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="acos.hst.mod">acos.hst.mod</A><tab to=T><TD>
6235<BR><TD> How to get Speed out of your HST and HST Dual Standard Modem on an
Apple IIGS
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="advdem.app">advdem.app</A><tab to=T><TD>
16645<BR><TD> Technical notes for Advanced DeMuffin II, a cracking tool
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="aecomman.app">aecomman.app</A><tab to=T><TD>
1792<BR><TD> A list of commands for Ascii Express
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="aids">aids</A><tab to=T><TD> 1024<BR><TD>
Method for detecting the "Cyberaids Virus", by The Chemist
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="alien.clues">alien.clues</A><tab to=T><TD>
1448<BR><TD> Passwords for Alien Mind, by The Undertaker and the Vandal
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ansi.spcs">ansi.spcs</A><tab to=T><TD>
24911<BR><TD> ANSI and VT100 Codes
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="apple.app">apple.app</A><tab to=T><TD>
4157<BR><TD> Combining Applesoft with Assembly Language
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="apple.txt">apple.txt</A><tab to=T><TD>
4189<BR><TD> The Text of the Apple-Microsoft Agreement
```

```
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="apple2.gs">apple2.gs</A><tab to=T><TD>
9388<BR><TD> The Sad, True Truth of the Apple II GS (Stands for Goddamned Slow)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="appleii.jok">appleii.jok</A><tab to=T><TD>
1384<BR><TD> The Unofficial Apple II Brainwash Test by Fred E. Long
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="applemaf.txt">applemaf.txt</A><tab to=T><TD>
22452<BR><TD> The Apple Mafia Story, as Told to Red Ghost
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="applenet.app">applenet.app</A><tab to=T><TD>
4096<BR><TD> Advertisement for Apple-net software. Note feature list
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="apples.txt">apples.txt</A><tab to=T><TD>
8230<BR><TD> Why the Apple II is Broken
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="appleser.app">appleser.app</A><tab to=T><TD>
11205<BR><TD> Apple //c Serial Port Information
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="applesoft.tips">applesoft.tips</A><tab
to=T><TD> 2320<BR><TD> The Beagle Brothers Applesoft Tips Guide
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="appswitc.app">appswitc.app</A><tab to=T><TD>
2677<BR><TD> Apple //e Soft Switch, Status, and other I/O locations
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="bin.ii">bin.ii</A><tab to=T><TD>
18944<BR><TD> Apple II Binary File Format, developed by Gary B. Little
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="bitsbaud.doc">bitsbaud.doc</A><tab to=T><TD>
11553<BR><TD> Bits, Baud Rate, and BPS, by michael A. Banks, 1988
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="bootl-6">bootl-6</A><tab to=T><TD>
102420<BR><TD> Collection of Apple-0riented Texts and Flotsam from the Early
1980's.
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="bootl-6.hac">bootl-6.hac</A><tab to=T><TD>
102420<BR><TD> Bootlegger Magazine Excerpts (Apple II Stuff)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="catfur.app">catfur.app</A><tab to=T><TD>
7176<BR><TD> Bit Blaster's Information on the Cat Fur Modem
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="catstuff.app">catstuff.app</A><tab to=T><TD>
9818<BR><TD> Expanding your Apple Cat // by the Warewolf
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cheat.app">cheat.app</A><tab to=T><TD>
4424<BR><TD> All manner of cheats for various Apple II games
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cheats">cheats</A><tab to=T><TD>
7416<BR><TD> LARGE Collection of Apple Cheats (Break into Monitor and Modify)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cheats.app">cheats.app</A><tab to=T><TD>
2749<BR><TD> The Penguin's Apple Cheats
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cheats2.app">cheats2.app</A><tab to=T><TD>
4498<BR><TD> Apple Pirate's Cheats
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="copyprog.app">copyprog.app</A><tab to=T><TD>
2991<BR><TD> How to Copy Programs, by the Three Musketeers
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="copyprot.app">copyprot.app</A><tab to=T><TD>
15163<BR><TD> Copy-Protecting your own disks, by Thomas T. Brylinski
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="correct.app">correct.app</A><tab to=T><TD>
5716<BR><TD> Corrections to programming for the Apple Cat
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cr.adder">cr.adder</A><tab to=T><TD>
1441<BR><TD> How to add Carriage Returns to Appleworks Databases
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crack1.txt">crack1.txt</A><tab to=T><TD>
1023<BR><TD> Introduction to a Talk on Software Piracy
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crackdos.app">crackdos.app</A><tab to=T><TD>
15403<BR><TD> Introduction to how AppleDOS operates
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crackin.app">crackin.app</A><tab to=T><TD>
9989<BR><TD> An introduction to cracking by The Necromancer
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crakowit.app">crakowit.app</A><tab to=T><TD>
3647<BR><TD> Kracowicz' Kracking Corner IV
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cramit.app">cramit.app</A><tab to=T><TD>
5062<BR><TD> An Introduction to Program Compression
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="cramit.txt">cramit.txt</A><tab to=T><TD>
7040<BR><TD> Some Tips on Cramming Data with an Apple
```

```
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crammin.app">crammin.app</A><tab to=T><TD>
5071<BR><TD> A simple compression scheme
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="crisis.app">crisis.app</A><tab to=T><TD>
1900<BR><TD> How to crack Crisis Mountain, by Doctor Who
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="deathcheat">deathcheat</A><tab to=T><TD>
517<BR><TD> Cheat for "Death Sword"
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="diskgo.txt">diskgo.txt</A><tab to=T><TD>
613<BR><TD> Getting Faster Apple DOS Speeds by Tamerlane of the Ring
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="diskjock.app">diskjock.app</A><tab to=T><TD>
51504<BR><TD> Examining protected Applesoft programs, by the Disk Jockey
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="dos.chart">dos.chart</A><tab to=T><TD>
1678<BR><TD> The DOS 3.3 Memory Access Chart
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="dosless.txt">dosless.txt</A><tab to=T><TD>
1792<BR><TD> Creating an Apple DOS-Less Disk
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="emu.pt.update">emu.pt.update</A><tab
to=T><TD> 3739<BR><TD> Message: Bugs in IIGS Proterm v1.9p
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="errors.app">errors.app</A><tab to=T><TD>
4286<BR><TD> A comment on error traps, by Nick Fotheringham
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="errors.txt">errors.txt</A><tab to=T><TD>
4480<BR><TD> A Comment on Error Traps by Nick Fotheringham from the Apple Barrel
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="expandca.app">expandca.app</A><tab to=T><TD>
9367<BR><TD> Expanding your Apple Cat, by Warewolf
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="futrae.app">futrae.app</A><tab to=T><TD>
4684<BR><TD> The Future Evolution of Ascii Express (Humor)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="icon.convert">icon.convert</A><tab to=T><TD>
3308<BR><TD> Converting Apple IIGS Icons to Clip Art by Marty Knight
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="iigsprob.hum">iigsprob.hum</A><tab to=T><TD>
2680<BR><TD> The Apple IIgs Sound Problem
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="joystick.app">joystick.app</A><tab to=T><TD>
5961<BR><TD> The Official Joystick Review Guide, by The Tracker
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="kickmacr.app">kickmacr.app</A><tab to=T><TD>
9981<BR><TD> How to kick butt with AE Macro Action
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krack1.app">krack1.app</A><tab to=T><TD>
2927<BR><TD> High Technology's Cracking Tutorial, Part I
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krack2.app">krack2.app</A><tab to=T><TD>
1765<BR><TD> High Technology's Cracking Tutorial, Part II
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krack3.app">krack3.app</A><tab to=T><TD>
2239<BR><TD> High Technology's Cracking Tutorial, Part III
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krack4.app">krack4.app</A><tab to=T><TD>
1887<BR><TD> High Technology's Cracking Tutorial, Part IV
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krack5.app">krack5.app</A><tab to=T><TD>
2560<BR><TD> High Technology's Cracking Tutorial, Part V
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krakowic.txt">krakowic.txt</A><tab to=T><TD>
13198<BR><TD> Kracowicz' Cracking Tips from ROM Radier
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="krckwczt.app">krckwczt.app</A><tab to=T><TD>
137510<BR><TD> The Kracowicz Basics of Cracking Series. A++
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="mac2info.app">mac2info.app</A><tab to=T><TD>
11449<BR><TD> Late-breaking (1987) information on The Macintosh II
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="maccrack.app">maccrack.app</A><tab to=T><TD>
5981<BR><TD> The Byte's introduction to Mac Cracking
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="machine.app">machine.app</A><tab to=T><TD>
13084<BR><TD> Black Bag's Introduction to Machine Language for Cracking
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="machinel.app">machinel.app</A><tab to=T><TD>
15408<BR><TD> Dr. Firmware's Tutorial of Machine Language
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="macteam.app">macteam.app</A><tab to=T><TD>
9569<BR><TD> Macteam's thoughts on copy protection on the Macintosh
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="memory.txt">memory.txt</A><tab to=T><TD>
12020<BR><TD> An Apple Peek Poke, Call List
```

```
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="miffins2.txt">miffins2.txt</A><tab to=T><TD>
1421<BR><TD> How to use Demuffin Plus
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.i">ml.part.i</A><tab to=T><TD>
5680<BR><TD> The Machine Language Tutorial Disk by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.ii">ml.part.ii</A><tab to=T><TD>
5370<BR><TD> The Machine Language Tutorial Disk Part II by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.iii">ml.part.iii</A><tab to=T><TD>
5627<BR><TD> The Machine Language Tutorial Disk Part III by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.iv">ml.part.iv</A><tab to=T><TD>
4970<BR><TD> The Machine Language Tutorial Disk Part IV by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.v">ml.part.v</A><tab to=T><TD>
5703<BR><TD> The Machine Language Tutorial Disk Part V by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ml.part.vi">ml.part.vi</A><tab to=T><TD>
5210<BR><TD> The Machine Language Tutorial Disk Part VI by Dr. Firmware
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="oneguy.txt">oneguy.txt</A><tab to=T><TD>
1408<BR><TD> Hey, If You Pirate the Game, Don't Call Tech Support
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="oo.world.info">oo.world.info</A><tab
to=T><TD> 3206<BR><TD> The Magnet Previews Out of This World GS
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="opcodez.app">opcodez.app</A><tab to=T><TD>
2811<BR><TD> Various Apple Opcodes
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="param2.app">param2.app</A><tab to=T><TD>
16201<BR><TD> Parameters of Nibbles Away II for various software packages
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peekpoke.app">peekpoke.app</A><tab to=T><TD>
21120<BR><TD> A really large collection of Apple II PEEKs and POKEs
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peeks.pokes">peeks.pokes</A><tab to=T><TD>
2957<BR><TD> Description of the differences between CALL, PEEK and POKE in
Applesoft
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peeks.pokes.1">peeks.pokes.1</A><tab
to=T><TD> 6166<BR><TD> Collection of Apple Peeks and Pokes
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peeks.pokes.2">peeks.pokes.2</A><tab
to=T><TD> 4396<BR><TD> Collection of Apple Peeks and Pokes in the Zero Page Area
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peeks.pokes.3.1">peeks.pokes.3.1</A><tab
to=T><TD> 14869<BR><TD> Apple Peeks, Pokes and Calls List Version 2.1 by The
Enforcer (May 1984)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="peeks.pokes.3.2">peeks.pokes.3.2</A><tab
to=T><TD> 5377<BR><TD> Miscellaneous Applesoft Information, by Control Reset
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="pitfall2.txt">pitfall2.txt</A><tab to=T><TD>
2176<BR><TD> Soft Docs for Pitfall 2: Lost Caverns
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="pm2600.app">pm2600.app</A><tab to=T><TD>
3045<BR><TD> The Poor Man's 2600 Hertz by Sir Briggs
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="pokelist.app">pokelist.app</A><tab to=T><TD>
19769<BR><TD> A really large collection of Apple II PEEKs and POKEs (Duplicate)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="quick.draw.3">quick.draw.3</A><tab to=T><TD>
5122<BR><TD> Quick-Draw Adventure Mapper by Sherlock Apple (Part III)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="quick.spells">quick.spells</A><tab to=T><TD>
3256<BR><TD> Quick-Draw Adventure Mapper by Sherlock Apple (Spells)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="secretk.app">secretk.app</A><tab to=T><TD>
6956<BR><TD> Secret Keys: Little easter eggs and news about Apple II games
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="softkey">softkey</A><tab to=T><TD>
21083<BR><TD> Softkey Unprotections for a Variety of Commercial Programs
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="trace2.app">trace2.app</A><tab to=T><TD>
11562<BR><TD> Mr. Xerox' boot tracing, volume I (badly converted)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="usr.16.8k">usr.16.8k</A><tab to=T><TD>
85773<BR><TD> The Info File on the USR Robotics 16.8k Model
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="vidomac.app">vidomac.app</A><tab to=T><TD>
33057<BR><TD> 1986 Seminar on "Macintosh in Film and TV Production"
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="vt100">vt100</A><tab to=T><TD> 3685<BR><TD>
DEC VT-100 Compatible Cursor Command Sequences
```

```
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="wings.fury.cht">wings.fury.cht</A><tab
to=T><TD> 606<BR><TD> Cheat to Wings of Fure
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="wizardry.4.info">wizardry.4.info</A><tab
to=T><TD> 3012<BR><TD> Advice about playing Wizardry IV
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="xmodem">xmodem</A><tab to=T><TD>
21581<BR><TD> XMODEM Protocol Reference, by Ward Christensen January 1, 1982
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="ymodem.s">ymodem.s</A><tab to=T><TD>
13048<BR><TD> YMODEM Source Code for GBBS by Mike Golazewski or Greg Schaefer
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="zmodem.gbbs">zmodem.gbbs</A><tab to=T><TD>
7045<BR><TD> The Addition of ZMODEM to GBBS!
</TABLE><P><TABLE WIDTH=100%><TR><TD ALIGN=RIGHT><SMALL>There are 98 files for a
total of 1,155,472 bytes.</SMALL><TR><TD ALIGN=RIGHT><SMALL>There are 3
directories.</SMALL></TABLE></BODY>
</HTML>
```

```
==============================================================================
DOCUMENT joystick.app
==============================================================================

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%/                            \%
%  The Official Joystik Review Guide  %
%         By: The Tracker         %
%                              %
%      A Rebel Alliance TextFile      %
%      Written: 12/8/85  1:45 PM      %
%                              %
%      Call RAPS> 1-206-584-6900      %
%\                            /%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Excuse me? You say that your 3 yr old joystick just took a dive? Your favorite game suddenly dosen't work and you realize you have a GOOD copy? Then it hits you: time to dump $30-$65 into a new JOYSTICK.

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Since the above situation just happened to me very recently, and I had to wade through tons of junk wares to the junk joystick section and was literally ATTACKED by salesmen trying to get me to buy a good joystick, I'm going to list some of the more popular joysticks in this file, as well as some other alternatives to sticks that you might consider purchasing.

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

```
             Joysticks
             ---------

TG Products, Inc.
"TG joystick", around $45-55 at good
             stores and larger soft-
             ware houses
```

This joystick is made by an older company, and as far as I know the first one. Before my TG crashed, I had it for 4 years (i.e. it's a rugged little baby) but once I needed to send it back because my button #0 kept sticking (the company charged my $10 and sent me a whole new stick, the kind with a round cable instead of a ribbon cable). On the bad side, TG wants major $$$ for one of them, and they do take some getting used to. One quick note: the new- er TG sticks do have a flip/flop switch to change to and from self-centering. On a scale of one to ten, TG sticks get a 7.

```
Kraft Inc. (not the cheese place)
"Kraft joystick", roughly $40.  I have
              not checked the price
              on these lately.
```

Kraft makes a nice, long-handled stick with the two firebuttons in some odd places (one on top and the other on the back...but oddly, it's very comfor- table). I believe they now have one with little buttons to switch from self centering to non-self centering. Once again, I am not familiar with this stick very much. Overall, from what I have seen, they are a good 8.

CH products
"Mach][" and "Mach III" joysticks, $45
                   and $55 re-
                   spectivly.

After my TG crashed, I went over to a Mach III and so far have not been disappointed. The difference between the two (besides $10) is that the III has a button #0 on the top of the han- dle while the][does not. To tell you the truth, if they had any]['s left I would have saved $10 and gone with it. Both have slide-switches to turn on and off self-centering and knobs (yes, knobs) to change the x and y deviation from the original axis (i.e. the little slidy things on a T.G. and most other joysticks). The Mach III might be rated down because the handle is kind of stubby (with the button on top) and too large for fingers but too small for hands. I suppose all sticks take get- ting used to though. Overall, let's give them an 8. So far the sticks have been good, eh?

Apple Computer Inc.
"Apple joystick", $60-$65.

BLEECH! This joystick is SUCKY! Yes, that's right, it SUCKS! Nothing against Apple, hell they make great computers, but the joystick need some HELP. The thing falls apart quickly (so I have noticed from many people) and most also say that it isin't very respondent (i.e. you move left and it says 'huh? left? ohhh left! duhhh'). The buttons are nice, I think. While the Kraft and CH products both just go down and stop (nice but 'dead') and the TG's have a bad habit of no stop or very little (that's why the buttons are first to go), the Apple has a nice, loud CLICK when you press one of the buttons down. To tell you the truth, you have my warning not to get one. Overall, a 6. (7 1/2 if they didn't fall apart so quickly...)

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Some Alternatives

Now that I have covered joysticks, let me just touch some quick points on other products.

```
TG paddles- I got a pair for $10.  Nice
          if you like a game that
          requires paddles, and fun
          to open and customize.

Sirus Joyport- Plug Atari-style sticks
             into your Apple.  There
             are a few things you
             should know though.
             1. Not made anymore.
             2. Requires special
              programming (if the
              game dosen't actually
              say 'joyport' then it
              won't work with it.)
             You can, however, plug
             in 2 Atari Sticks and 2
             regular sticks (or a
             stick and a paddle!)
             and choose with a small
             switch on the top of the
             unit.

Wico joystick Adapter- Same as above
                   but no special
                   programming!
                   But, for $20 it
                   still dosen't
                   work with every-
                   thing.

TG Trakball- I think we all know what
           a trackball is, if you
           don't, imagine an upside-
           down mouse that sits in
           one spot and you roll it's
           ball(s?).    Let me say that
           it's no good unless you
           live for games that re-
           quired them (Centipede on
           the Apple is very easy
           with it).    For $70 I say
           forget it.

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

I am sure that I missed a few sticks and probably a few 'alter- natives' but those are the main ones. If you have any comments, feel free to call RAPS (the number is on the top) and leave me feedback. Or leave me e-mail on Apple Manor [716-654-POOF]. Thanks for reading and have a nice day.

------------- The Tracker -------------

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

ALL POINTS BULLETIN: If you have a ware called GRABBIT (9 months old as of 12/8/85, and never protected) please call RAPS [206-584-6900] and upload it. Thank you very much.

```
================================================================================
DOCUMENT kickmacr.app
================================================================================

(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\
(\                                                          (\
(\            How to kick butt with AE macro action           (\
(\              Written by: The Radioactive Snail                   (\
(\                                                          (\
(\     A continuation of: How to kick butt with AE cursor action      (\
(\                Written by: [mr. sandman]                   (\
(\                                                          (\
(\ The Last Dimension AE ..............................[10meg] 214/827-5249 (\
(\                                                          (\
(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\
```

The Macro Library File
=====================

I'm sure most of you have done some screwing around with the MACRO.LIB file, and know most of the details about it, so I will just skim, over that here:

```
      The macro library (MACRO.LIB) contains the folowwing data:

            line#0      <blank>
            line#1+ macro character/displayed name/filename of macro.
```

Note that the first line of the MACRO.LIB file is kept blank, this is necessary for use. The "macro character" is the character you will type to get the given macro, the "displayed name" is the name by which AE calls the macro, "filename of macro" is simply the filename of the macro (minus the .MAC suffix). Therefore if line #1 of your macro library looked like this:

```
a/The Snake's Den/SNAKE
```

then from the -> or +> prompt, you can press return and AE will say:

```
Select? (A-Z,/,?)
```

if you press A (capitol or lowercase) AE will look for the macro beginning with "a/". Finding the Snake's Den macro, it will spin the drive, loading the file called "SNAKE" and say:

```
The Snake's Den <macro loaded>
```

If you have the "dial after loading macro" option set in the install program (on system menu #5 I think) it will then dial the macro for you. Note: The [Y]editor from the AE main menu is an easy way to modify the MACRO.LIB file.

From the "Select? (A-Z,/,?)" prompt, pressing "/" will load the MACRO.LIB file and present you with a list of the current library. Pressing "?" will give you a list of macros (#0 to #;) currently loaded.

When a macro is loaded, you can dial it by pressing "Dial: m".   M for macro.

The Macro Editor
================

Now comes the real kick-ass stuff: Pressing [U] from the main menu loads the macro editor from disk, and allows you to load/save/edit macros. The first page of options you are presented with is fairly self-explanitory. It is simply a list of default options for the macro when it is loaded. When changing the phone number, several extra characters can be added:

Phone#: xxx-xxxx

If you place a / before the phone number, it will autodial the macro instead of just dialing it once. Example:

Phone#: /xxx-xxxx

If you place a !n before the phone number, it will (upon connect) execute macro #n. Example

Phone#: xxx-xxxx !0

This would, upon connect, execute macro #0. Note that both the autodial and the execute upon connect can be used at the same time:

Phone#: /xxx-xxxx !0

Pressing [D] from the macro menu will take you to the display-edit macros screen

Display-edit Macros

Here you are allowed to change 12 different macros (#0 to #;) to whatever you want. Again, there are special characters:

```
            Delay ............................ *
            AE command character ............. \
            Slow ............................. ?
            String handshaking ............ <xxx>
            Handshake ........................ %
            Wildcard handshake ............... =
            Literal .......................... @
            Conditional handshake ............ ~
                Carrige return ................... '
            Jump to new macro line ........... \L
```

Note: these are the default values, and can be changed from the "C" option
      from the install program.

Delay

When excuting a macro, if AE encounters a delay character, it will pause for 1/2 second (ie. "****" would produce a two second delay).

AE Command Character

When AE encounters this character in a macro, it interprets it as if you had hit your terminal escape key, then the character following the command character.

```
      Slow
      ====
```

When placed at the beginning of a macro, AE will excute the macro line at 1/3 normal speed (usefull for systems with spinning cursors and no input buffer.. yeech).

```
      String handshake
      ================
```

There are actually two separate characters for the conditional handshake, a begining character, and an ending character (usually a set of one of the three brackets). If AE finds the beginning character for the conditional handshake, it reads all the text until it encounters the ending character, the AE waits for the other computer to send the EXACT string contained in the brackets until proceeding. Example:

```
<pukenuke>..rest of macro string..
```

This would pause until the string "pukenuke" was recieved over the line, then it would continue with the rest of the macro.

```
      Handshake
      =========
```

This is quicker and easier than the conditional handshake, but at some times it will not quite work right for a certian purpose. When AE finds the handshake character, it waits for the remote computer to send the character immediatly following the handshake character, for example:

```
%:nuke'em
```

Would wait for a ":" to be sent over the modem, then print "nuke'em".

```
      Wildcard handshake
      ==================
```

When encountering this character, AE will wait until a character comes over the modem, it does not matter what character it is, AE will just wait until one does. For example:

```
=ugamugawuga.
```

Would wait for ANY character to be recieved, then print "ugumugawuga"

```
      Literal
      =======
```

If you wish to send a macro command character (the * for instance, which usually produces a delay) insert this character before it. Example:

```
@* yer screwed @*
```

would print:

```
* yer screwed *
```

instead of:

<pause> yer screwed <pause>

Conditional handshake

This waits for a certian character (like the normal "%" handshake) and then waits for the next character and either 1) continues with the current macro or 2) aborts the current macro, and jumps to another one. Probably the best (and only?) use of this is for reading mail on a BBS system. If a system said either:

You have mail waiting!

or:

Sorry, you have no mail waiting.

You could make a macro like:

<ou have>~ mn1<continued macro string to read mail..>

This macro would wait for the string "ou have" (because one you starts with a capitol Y, the other does'nt), then it would wait for a space (the next character regardless) then if the next character was an "m" (as in "mail waiting") it will skip the "n1" part and go to the <continued macro string..> part and read the mail. If the next character is an "n" (as in "no mail waiting") it will abort the current macro and jump to macro #1.

Carrige return

Because it is advised that you take up only one or two macros for logon procedure (to leave room for the creative ones later), sometimes it is necessary to enter a carrige return (after a password for example), all this character will do is enter a <CR>. There is automaticly a <CR> after the end of every macro, if you put this carrige return character at the END of a line, it will abort the usual carrige return. Therefore:

pukenuke'''

Would only give you:

pukenuke<CR><CR>

Jump to new macro line number

A backslash (the AE command character) followed by L, then a number (or :,;) will jump to the new macro line number.

Full example, and uses of macros

Say your name was "PUKENUKE" and your password to "the global war BBS" was "NUKE'EM", the phone number was: 999-9999 and that the logon procedure looked like:

```
Welcome to the global war BBS,
Your local nukefull system.
Sysop: Lord Nuke

Enter username: PUKENUKE<CR>
Enter password: NUKE'EM<CR>

Searching..found ya.

Welcome PUKENUKE, today is march 12, 2093.
Apacolypse wow!

Press <RETURN> to enter system: <CR>
```

You would make the macro lib file to read:

```
a/The Global War BBS/NUKE
```

Load the macro editor, change the phone# to:

```
Phone#: /999-9999 !0
```

So it autodialed upon loading, and executed macro #0 when it connected, then you changed macro #0 to:

```
#0 <username: >PUKENUKE'<password: >NUKE@'EM<system: >
```

This would log you on, and automaticly take you into the system.

Now for the rest of the macros

So, what do ya do with all the other macros ya say? Well, make them into your favorite sayings:

```
#0 logon macro+mail read
#1 logon w/no mail
#2 rah.
#3 When the going gets tough, the smart run like hell.
#4 Pukenuke.
#5 -=< The PUKENUKE >=-
#6 How un-nukefull of you.
#7 Go nuke yer mamma.
#8 Go commit nukeacide.
#9 Nuke or be nuked.
#: I think not.
#; Nukin' some ass.
```

Now, to display all those "nuke" messages, and signoff macros (like #5 and #4) you have to do a ^W (again depending on the install program) and the macro number.

So at the end of every message, you could do ^W5 which would print:

```
-=< The PUKENUKE >=-<CR>

And in chat, if a sysop told you that you were a complete asshole, you could
do this:

^W7^QHY<CR>

which would do this:

Go nuke yer mamma<CR>
+>Disconnect? Yes!
[click]

(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\

How to kick butt with AE macro action has been a presentaion of TP&the
Heartbreakers.    Typed and figured out by: The Radioactive Snail.  Credits
to [mr. sandman].

(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\(\
================================================================================
DOCUMENT krack1.app
================================================================================

                  *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
                  :*                        *:
                  *:    High Technology's        :*
                  :*    Cracking Tutorial  *:
                  *:                        :*
                  :*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:

                           Written by:
                  High Technology & Sherlock Apple!
=-------------------------------------------------------------------------------=
```

Written for: Sherwood Forest][and Sherwood Forest ///

```
   _______
  /       \
 < Preface >
  \_______/
```

   This series is aimed to help you de-protect certain programs to make back-up copies of the programs hereafter, and only for that purpose. The authors take no responsibility for an(y) illegal cop(y)(ies) made by the end user of this information, nor any damage to programs, hardware, or any other physical damage done by the use of the information hereafter.

   The authors urge you to attempt to make nibble copies of the program before attempting any modification to the program. The techniques described in this series may not work on all versions of the program. In most cases there are many other ways to de-protect the program (such as nibble counts) and if any of you know of a better way please let us know.

   More advanced crackers or programmers who know machine language pretty well might want to skip the text and just read the => prompts. The -> prompt indicates a place in the instructions where you may not need to go any further.

=---=

This weeks topic: Xerox educational games

=---

Xerox is known for their excelence in education games such as the Stickybear series, Chivalry, Fat City, Pic Builder, and others. The protection on these games is fairly standard, a single nibble count. To modify the nibble count so it does not function, we need a sector editor. Inspector is the most popular, although the sector editors in Nibbles Away][and Copy][+ both are fine. Of course, we suggest you make a back-up of the program before modifing any data.

=> Read Track 2, Sector 6
=> Change byte $00 from $A9 to $60
=> Write Track 2, Sector 6 back to disk.

Notes: You just disabled the nibble count. The start of the routine was at, of course, Track 2, Sector 6. The routine was in machine, so $A9 (the first byte of the routine) stood for LDA (LoaD Accumulator). By replacing it with $60 (which stands for RTS, or ReTurn from Subroutine) we returned to the main program without doing the nibble count. This will not work for ALL of the Xerox software, such as Stickybear Bop and Pic Builder, but will work for almost all of it.

```
=------------------------------------------------------------------------------=
  _________________
 |             |
 | Happy Cracking! |
 | High Technology |
 | The Apple Mafia |
 |_________________|
```

- End of File -

```
==============================================================================
DOCUMENT krack2.app
==============================================================================

                  *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
                  :*                        *:
                  *:    High Technology's        :*
                  :*    Cracking Tutorial  *:
                  *:                        :*
                  :*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:

                           Written by:
                  High Technology & Sherlock Apple!

Written for: Sherwood Forest ][ and Sherwood Forest ///

=------------------------------------------------------------------------------=

This weeks topic: Homeword

=------------------------------------------------------------------------------=

   Homeword, by On-line systems, is a fantastic word processor, which is both
well doccumented and easy to use. It is the first word processor, to my
knowledge, to incorporate icons at all menu prompts. It comes with a cassette
which helps in teaching you how to use the program. The de-protection is fairly
simple, and like the Xerox series, requires only a sector editor. If you do
not know how to use the Inspector sector editor, the docs can be found on
Sherwood Forest ///.

=> Read Track 10, Sector A.
=> Change byte $00 from $CE to $60
=> Write Track 10, Sector A back to disk

This does the exact same thing as the Xerox de-protecting. Note that $CE stands
for DEC (DECrement accumulator). By replacing it with $60, RTS (ReTurn from
Subroutine), you are replacing the first byte of the nibble count routine, and
telling it to jump back to the main program, without executing the nibble count
at all. If the nibble count were to be executed the program would crash,
re-boot, or freeze up.

=------------------------------------------------------------------------------=


  _________________
 |             |
 | Happy Cracking! |
 | High Technology |
 | The Apple Mafia |
 |_________________|

- End of File -
```

```
================================================================================
DOCUMENT krack3.app
================================================================================

                  *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
                  :*                        *:
                  *:    High Technology's        :*
                  :*    Cracking Tutorial  *:
                  *:                        :*
                  :*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:

                           Written by:
                  High Technology & Sherlock Apple!

Written for: Sherwood Forest ][ and Sherwood Forest ///

=-------------------------------------------------------------------------------=
This weeks topic: Mr. Cool
=-------------------------------------------------------------------------------=

   Mr. Cool is a 3-dimensional Q-Bert type game that has copy protection differ
ent that the others described in this tutorial so far.

=> Boot your DOS 3.3 system master and insert a blank disk
=> Type INIT HELLO. When you get the ] or > prompt, type DELETE HELLO.
=> Insert Mr. Cool disk and type "BRUN MRCOOL"
=> When the picture comes up, hit [ RESET ]. Type "CALL -151" to enter monitor.
=> Type 4000: EA EA EA (this removes the nibble count)
=> Type 8500: 60 (this removes the high score read)
=> Type 876C: 60 (this removes the high score write)
=> Insert the disk you just formatted and type "BSAVE MR. COOL,A$4000,L$5500"

Notes: The "4000: EA EA EA" is the start of the nibble count routine. Putting
"EA EA EA" in locations $4000-$4002 places the NOP (or No OPeration) code in
place of the jump to the nibble count. The "8500: 60" removes the high score
read routine. The routine starts at $8500 and by placing a $60 (ReTurn from
Subroutine) it never executes the routine and jumps back to the main progam.
"876C: 60" removes the high score write. It is the same as above. The reason
for disabling the high score functions is simple. If not disabled, when the
high socres are written to the disk, it would overwrite part of the MR. COOL
file or whatever else happens to be on the sectors that it uses to store the
names and scores. The read function is disabled because when it read the data
from the disk it woud crash because those aren't the names and scores.

=-------------------------------------------------------------------------------=
   _________________
  |              |
  | Happy Cracking! |
  | High Technology |
  | The Apple Mafia |
  |_________________|

- End of File -
```

```
==============================================================================
DOCUMENT krack4.app
==============================================================================

                  *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
                  :*                        *:
                  *:    High Technology's        :*
                  :*    Cracking Tutorial   *:
                  *:                        :*
                  :*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:

                           Written by:
                  High Technology & Sherlock Apple!

Written for: Sherwood Forest ][ and Sherwood Forest ///

=-----------------------------------------------------------------------------=
This weeks topic: Print Shop
=-----------------------------------------------------------------------------=
```

  Print shop, by Broderbund Software, was well written by its two authors, David Balsam and Martin Kahn. It makes terrific banners, signs, letter heads, among other things and the doccumentation is excellent. The docs were typed by Dr. Vax and can be found on Apple Manor. The de-protection is more complicated than any other we have yet seen.

```
=> Use Disk Editor 2.0 to search for (in hex) 20 16 70
=> Replace (with a sector editor) all mentioned locations containing 20 16 70
   (approx. 15) in them to EA EA EA.
=> Replace (with a sector editor) all mentioned locations containing 20 F9 77
   (approx. 2) in them to EA EA EA.
```

Notes: "20 16 70" is the machine language code for Jump to SubRoutine to $7016, which is the beginning of a nibble count. "20 F9 77" is the machine language code for Jump to SubRoutine $77F9 which is the beginning of a routine that is a fake nibble count which jumps to the real nibble count. By replacing them with "EA EA EA" standing for NOP or No OPeration, we eliminate the jumps to these locations and therefore never execute the nibble counts, making the program de-protected.

```
=-----------------------------------------------------------------------------=


  _________________
 |             |
 | Happy Cracking! |
 | High Technology |
 | The Apple Mafia |
 |_________________|

- End of File -
```

```
==============================================================================
DOCUMENT krack5.app
==============================================================================

                  *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
                  :*                        *:
                  *:    High Technology's        :*
                  :*    Cracking Tutorial  *:
                  *:                        :*
                  :*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:

                           Written by:
                  High Technology & Sherlock Apple!

Written for: Sherwood Forest ][ and Sherwood Forest ///

=-------------------------------------------------------------------------------=
This weeks topic: Electronic Arts (EOA)
=-------------------------------------------------------------------------------=

   Electronic Arts, are the makers of One On One, Last Gladiator, Archon, Cut
and Paste, and many other excelent products. Their protection scheme is fairly
standard and the text below will mork for most of their stuff.

=> Boot the DOS 3.3 System Master. When you get the ] prompt, type INIT HELLO.

=> Next, insert your Advanced Demuffin disk in the drive. Type "BRUN ADVANCED
   DEMUFFIN 1.1"

=> Type CALL -151 to enter monitor then 3 machine language codes:
    [1] B8F1: BB  [ RETURN ]  [2] B8FC: CF  [ RETURN ]      [3] 803G  [ RETURN ]

=> Boot up a sector editor and read Track $20, Sector $0F. Write it onto Track
   $03, Sector $00.

=> Now use a bit copy program (such as EDD or Locksmith) to copy tracks 0-2 of
   the origional disk to your converted disk.

     This converts the disk to standard DOS 3.3, except the RWTS...lets take
     care of that right now...

=> Now use a sector editor to read Track $02, Sector $03. Change bytes:

   [1] $47 from $BB to $BA  [2] $51 from $CF to $AD

     This changes the RWTS checksums on the disk to standard DOS 3.3 RWTS.

=> Now read Track $01, Sector $0F

=> Change bytes $68-$6A from 20 A2 A1 to 18 60 EB

     This removes the nibble count.

-> If you are working on Cut & Paste or Last Gladiator stop here <-

=> One On One and others: Read Track $0C, Sector $04. Change bytes $06-$08 from
   A0 18 88 to 18 60 C8. Changes bytes $DC and $DD from A0 FF to 18 60. Read
```

Track $09, Sector $02. Change byte $1F from $01 to $FD.

You removed the secondary nibble count found in most EOA games except Last Gladiator and Cut & Paste. One On One is the example used here. If this last ni bble count is not removed, the game will function normally but the players heads will spin (How sneaky!!!).

=--=

```
  _________________
 |              |
 | Happy Cracking! |
 | High Technology |
 | The Apple Mafia |
 |_________________|
```

- End of File -

```
==============================================================================
DOCUMENT krakowic.txt
==============================================================================
```

```
               ***************************************
               *   KRACKOWITZ'S CRACKING TIPS      *
               ***************************************
                    FROM: THE ROM RAIDER
                        DR. DIGITAL

                CALL HER MAJESTY'S SECRET SERVICE
                   3 0 3 - 7 5 1 - 2 0 6 3
```

ALONG WITH A NUMBER OF REQUESTS FOR MATERIAL USEFUL TO THOSE WHO ARE NOT YET IN THE RANKS OF PROFESSIONALS IN THIS FIELD, IT HAS BEEN POINTED OUT TO ME THAT I AM ALL TOO WILLING TO SUGGEST BURNING THIS PROM, INSTALLING THAT ROM, AND GENERALLY MAKING WHOLESALE HARDWARE CHANGES IN AN UNSUSPECTING APPLE, WITHOUT PROVIDING BACKGROUND INFORMATION FOR THE UP-AND-COMING KRACKISTS OF THE FUTURE.

THIS SERIES, WHILE AIMED AT THE BEGINNING TO INTERMEDIATE KRACKIST, WILL STILL ASSUME A REASONABLE KNOWLEDGE OF ASSEMBLY LANGUAGE. IF YOU FIND THESE DISCUSSIONS ARE STILL TOO HEAVY INTO MACHINE CODE FOR YOU, THEN IT'S BEST TO BUY A BOOK LIKE ROGER WAGNER'S "ASSEMBLY LINES" OR EQUIVALENT, AND STUDY IT CAREFULLY (IF, ON THE OTHER HAND, YOU FIND THAT THIS IS ALL BENEATH YOU, JUST KEEP A KNOWING SMIRK ON YOUR LIPS AS YOU SKIP LIGHTLY OVER THESE EPISODES - THERE MIGHT BE SOMETHING YOU MISSED BECAUSE YOU HAD A BAD HANGOVER ONE DAY IN KRACKING 101).

IN THIS AND FUTURE EPISODES IN THE 'BASICS OF KRACKING' SERIES, WE'LL DEAL WITH THE FUNDAMENTALS OF THE KRACKIST'S ART, STARTING WITH THE HOW (AND WHY) OF MAKING ALTERATIONS IN THE APPLE'S "PERMANENT" MEMORY. FIRST OF ALL, THE MOST IMPORTANT SINGLE TOOL AVAILABLE TO THE ASPIRING KRACKIST IS REPLACING THE AUTOSTART ROM ON THE MOTHER BOARD WITH AN "OLD MONITOR" ROM. WITH THIS ROM IN PLACE, YOU CAN HIT 'RESET' WHENEVER YOU WANT, AND ALWAYS BE RETURNED TO THE MONITOR FOR THE BEGINNING OF THE SNOOPING PROCESS. THIS CHANGE, INCIDENTALLY, WILL MAKE AVAILABLE TO YOU A REASONABLE SET OF "STEP AND TRACE" UTILITIES (SEE THE APPLE II REFERENCE MANUAL.   PP 51-53).

TO UNDERSTAND WHAT THE DIFFERENCES ARE BETWEEN THE TWO ROMS, LET'S TAKE A MINUTE TO EXAMINE WHAT PRESSING THE 'RESET' KEY DOES (OMIGOSH, MAUDE, THERE HE GOES AGAIN ON THAT DETAILED TECHNICAL CRAP!). INSTEAD OF GOING THROUGH THE KEYBOARD INPUT ROUTINE AT C000, THE RESET KEY IS CONNECTED DIRECTLY TO PIN 40 OF THE 6502 MICROPROCESSOR CHIP. WHEN THIS PIN IS CONNECTED TO GROUND (0 VOLTS), THE COMPUTER JUMPS UNCONDITIONALLY TO THE ADDRESS CONTAINED IN LOCATIONS FFFC AND FFFD. THIS IS NOT A TRUE INTERRUPT, SINCE THE APPLE FORGETS WHAT IT WAS DOING BEFORE THE LINE WAS "YANKED", BUT IT IS AN EXAMPLE OF 'VECTORING' OR SENDING THE COMPUTER TO A SPECIFIC PLACE BY SETTING AN ADDRESS INTO THE PROGRAM COUNTER. IN THE AUTOSTART ROM, THESE TWO LOCATIONS CONTAIN 62 FA, SO THE NEXT INSTRUCTION TO BE EXECUTED IS AT FA62.   THIS SERIES OF ROUTINES
(SEE P. 143 AND PP. 36-38 OF THE REFERENCE MANUAL) CHECKS TO SEE IF THE COMPUTER IS BEING POWERED UP FOR THE FIRST TIME (COLDSTART) OR RESET WITH THE POWER ON (WARMSTART). IF IT IS A WARMSTART, THE SYSTEM JUMPS TO THE INSTRUCTIONS AT LOCATIONS 3F2 AND 3F3, AND BEGINS RUNNING THE PROGRAM FOUND THERE (USUALLY BASIC AT E000).

THE "OLD MONITOR" ROM, HOWEVER, HAS 59 FF STORED IN FFFC-D. THIS CAUSES AN

APPLE II (OR A II+ WITH AN INTEGER CARD AND THE RED SWITCH "UP") TO GO TO ROUTINES WHICH SET UP THE KEYBOARD FOR INPUT, THE TV FOR OUTPUT, AND WIND UP IN THE MONITOR WITH THE '*' PROMPT DISPLAYED. IN CONTRAST TO THE AUTOSTART ROM, WHERE ANYONE CAN TELL THE RESET BUTTON WHERE TO GO, THERE IS NO WAY TO PREVENT A RESET FROM GOING TO FF59 AND WINDING UP IN THE MONITOR. THIS IS OBVIOUSLY ESSENTIAL IF YOU WANT TO BREAK INTO A GAME AND START EXAMINING THE CODE, BUT IT HAS ITS OWN SET OF PROBLEMS.

IN THE PROCESS OF SETTING UP THE I/O DESCRIBED ABOVE, ESPECIALLY IN SETTING UP THE TEXT WINDOW ON THE SCREEN, A NUMBER OF LOCATIONS IN ZERO PAGE MUST BE CHANGED. THE FOLLOWING LOCATIONS WILL PROBABLY BE ALTERED (ALL HEX): 20,21,22,23,24,25,28,29,32,33,35, 36,37,38,39, AND 48.   WORSE THAN THAT, THE ENTIRE SCREEN SCROLLS UP ONE LINE WHEN THE MONITOR PROMPT IS PRINTED, WHICH LOSES THE ENTIRE TOP ROW OF THE TEXT SCREEN (LOCATIONS 400-427), AND ALTERS THE CONTENTS OF ALL THE OTHER LOCATIONS FROM 400-7FF, WITH THE EXCEPTION OF THE "SCRATCHPAD" REGIONS AT 478-47F, 4F8-4FF, ETC. (THE COMPUTER WIMP AT YOUR SCHOOL SAYS THAT THE TOP LINE "FALLS INTO THE BIT BUCKET", BUT YOU KNOW HOW EVERYONE FEELS ABOUT HIM.)

AS MOST SOFTWARE PROTECTORS KNOW, THIS WILL KEEP MOST OF THE AMATEURS OUT OF THE PROGRAM, AND YOU'LL SEE EVIDENCE OF THIS TECHNIQUE IN THE FORM OF A LOT OF "GARBAGE" ON THE TEXT SCREEN WHEN YOU RESET OUT OF A PROTECTED GAME. OUR JOB, THEN, IS TO KEEP THESE ZERO PAGE AND SCREEN MEMORY LOCATIONS FROM BEING LOST, SINCE MOST PROTECTION SCHEMES USE THESE AREAS IN SOME WAY OR OTHER (BR0DERBUND, FOR EXAMPLE, HAS RECENTLY BEEN STORING THE ADDRESS MARKER FOR THE DISK TRACK IN LOCATIONS 20, 21, AND 22).

THE SAFE WAY TO PREVENT INFORMATION FROM BEING LOST FROM THESE "VOLATILE" LOCATIONS IS TO TRANSFER ALL OF THE CONTENTS TO A SAFE AREA -- LOCATIONS 2000 & UP (OR 4000 & UP) WHERE A HI-RES PICTURE NORMALLY RESIDES. IN FACT, IT WOULD BE BEST TO SAVE EVERYTHING FROM 0 TO 8FF, SINCE BOOTING A DISKETTE TO SAVE THE DATA ALSO DESTROYS LOCATIONS 800-8FF. (REMEMBER THE FIRST LAW OF DISK KRACKING - TRACK 0, SECTOR 0 ALWAYS STARTS WITH D5 AA 96 AND ALWAYS LOADS INTO 800-8FF). BECAUSE THIS IS THE BEGINNING CLASS, LET'S LOOK AT TWO EXAMPLES OF SHORT BINARY SUBROUTINES THAT WILL DO THE "SAVE" FOR US. BOTH START, AS WILL BE EXPLAINED LATER, AT LOCATION FECD IN THE F8 ROM.   THE FIRST IS THE MOST STRAIGHTFORWARD AND EASIST TO FOLLOW:

```
  LDY  #$00    ;CLEAR Y-REGISTER
  LDA  $00,Y   ;GET A BYTE FROM 0+Y
  STA  $2000,Y ;STORE AT 2000+Y
  LDA  $0100,Y ;THEN FROM 100+Y
  STA  $2100,Y ;TO 2100+Y
  LDA  $0200,Y ;AND SO ON UNTIL
  STA  $2200,Y ;WE HAVE COVERED
  LDA  $0300,Y ;ALL THE MEMORY
  STA  $2300,Y ;'PAGES' FROM 0 TO 8
  LDA  $0400,Y ;AND STORED INTO
  STA  $2400,Y ;PAGES 20 TO 28
  LDA  $0500,Y
  STA  $2500,Y
  LDA  $0600,Y
  LDA  $2600,Y
  LDA  $0700,Y
  STA  $2700,Y
  LDA  $0800,Y
  STA  $2800,Y
  INY        ;THEN ADD 1 TO Y-REG
  BNE  $FED0   ;AND REPEAT IF < 256
  JMP  $FF59   ;WHEN WE'RE ALL DONE
             ;JUMP TO MONITOR START
```

THIS 61-BYTE ROUTINE, IF IT COULD BE EXECUTED AUTOMATICALLY WHEN THE RESET KEY IS PRESSED, WOULD SAFELY STASH ALL OF THE CHANGEABLE MEMORY AND EXIT GRACEFULLY INTO THE MONITOR.

A MORE COMPACT AND GENERAL, BUT LESS OBVIOUS ROUTINE IS SHOWN BELOW. IT IS INCLUDED BECAUSE IT IS TYPICAL OF THE "MEMORY MOVE PROGRAMS" THAT WE WILL EVENTUALLY HAVE TO WRITE IN KRACKING ALMOST ANY PROGRAM.

```
  LDY  #$00    ;CLEAR Y-REGISTER
  LDA  $00,Y   ;XFER THE ZERO PAGE TO
  STA  $2000,Y ;2000-20FF SO WE CAN USE
  INY        ;THE ZERO PAGE MEMORY
  BNE  $FED0   ;FOR THE OTHER MOVES
  LDA  #$00    ;SET UP LOCNS 0 & 1 AS A
  STA  $00     ;2-BYTE POINTER FOR THE
  STA  $02     ;SOURCE ADDRESS, USE 2&3
  LDA  #$01    ;AS 2-BYTE POINTER FOR
  STA  $01     ;THE DESTINATION ADDRESS
  LDA  #$21    ;STARTING AT $2100
  STA  $03
  LDA  ($00)<- ;GET A BYTE FROM 100-UP
  STA  ($02) ^ ;STORE AT 2100-UP
  INC  $02   ^ ;INCREMENT LO-ORDER BYTE
  INC  $00   ^ ;OF SOURCE & DESTINATION
  BNE  ->->->^ ;(BACK TO LDA ($00) IF
           ^ ;LO-ORDER IS <256
  INC  $03   ^ ;IF LO-ORDER=0, INC THE
  INC  $01   ^ ;HI BYTE OF EACH
  LDA  $01   ^ ;CHECK TO SEE IF HI-BYTE
  CMP  $#09  ^ ;IS 9 -WE'RE THRU AT 8FF
  BNE  ->->->^ ;IF NOT, LOOP BACK TO
             ;THE LOAD/STORE UNTIL
             ;WE'RE ALL DONE
  JMP  $FF59   ;EXIT THRU MONITOR
```

UNLIKE THE FIRST ROUTINE, THIS ONE (AT 47 BYTES) USES RAM LOCATIONS 0 THROUGH 3, SO THE ZERO PAGE MUST BE TRANSFERRED BEFORE IT IS ALTERED BY USING THOSE ADDRESSES AS POINTERS. WHILE THE FIRST ROUTINE MUST GROW BY SIX BYTES FOR EACH ADDITIONAL PAGE TRANSFERRED, THE SECOND NEEDS ONLY TO HAVE THE "9" IN THE COMPARE STATEMENT CHANGED TO THE APPROPRIATE VALUE ONE HIGHER THAN THE LAST PAGE NUMBER BEING TRANSFERRED.

TO RETURN TO THE BUSINESS OF ALTERING ROMS, IT IS EASY TO SEE THAT AN AUTOSTART ROM COULD BE MADE TO BEHAVE LIKE AN OLD ROM JUST BY CHANGING LOCATIONS FFFC-D TO 59 FF FROM 62 FA. (A NOTE TO THE FAINT-HEARTED--YOU CAN BUY AN OLD MONITOR F8 ROM FOR ABOUT $10 AND PLUG IT DIRECTLY INTO YOU APPLE'S F8 SOCKET, BUT YOU WON'T HAVE ALL THE BENEFITS WE'VE BEEN TALKING ABOUT). AS LONG AS WE'RE GOING TO THE EFFORT OF MAKING A CHANGE, THOUGH, WE MIGHT AS WELL ADD ONE OF THE ROUTINES ABOVE AND ALLOW THE NEW ROM TO SAVE THE VOLATILE MEMORY FOR US. TO DO THIS, WE'LL HAVE TO GIVE UP SOMETHING IN THE ROM, AND THE MOST EASILY SURRENDERED AREA FOR MOST OF US IS THE TAPE READ/SAVE ROUTINES AT $FECD. IF WE THEN CHANGED FFFC-D TO CD FE, THE MEMORY FROM 0 TO 8FF WOULD BE SAVED TO 2000-28FF EVERY TIME THE 'RESET' KEY WAS PRESSED. SINCE IT'S SOMETIMES INCONVENIENT TO HAVE THAT HAPPEN WHEN THE RESET KEY IS PRESSED, WE CAN REQUIRE

THAT A SPECIFIC KEY BE ALSO PRESSED TO MAKE IT OCCUR. THESE FEW INSTRUCTIONS INSERTED BEFORE EITHER OF THE ROUTINES ABOVE WILL GIVE A "RESET AND SAVE" WHEN THE "-" KEY IS HELD DOWN (OR WAS THE LAST KEY PRESSED), WHILE GIVING A REGULAR "OLD RESET" THE REST OF THE TIME.

```
  LDA  $C000  ;LOOK AT THE KEYBOARD
  ROL        ;MASK OFF HIGH BIT
  CMP  #$5A   ;WAS IT "-"?($2D X 2=$5A)
  BNE  ->->-> ;IF NOT, BRANCH TO THE
          ! ;LOCATION WITH THE
          ! ;"JUMP FF59" INSTRUCTION
          ! ;AT THE END OF THE SAVE
          ! ;SUBROUTINE.
```

OK, OK - WE ALL AGREE THAT THESE WOULD BE NEAT THINGS TO HAVE IN THE F8 ROM, SO HOW DO WE GET IT THERE? FIRST, GET HOLD OF A PROMBURNER (PROMBLASTER, EPROM PROGRAMMER, ETC.) THAT WILL PROGRAM 2716 EPROMS. EACH ONE IS DIFFERENT, SO I WON'T TRY TO GIVE DETAILED INSTRUCTIONS ON THE ACTUAL PROGRAMMING. BUY OR BORROW A FRIEND'S OLD F8 ROM (OR GET THE BINARY FILE) THEN TYPE IN OR LOAD IN THE CHANGES YOU WANT TO MAKE AT FECD & UP AND AT FFFC-D, AND PROGRAM A 2716 EPROM WITH OUR MODIFIED VERSION OF APPLE'S F8 MONITOR ROM.

ALL THAT REMAINS TO TAKE FULL ADVANTAGE OF THE NEW F8 ROM IS TO MAKE A SLIGHTLY MODIFIED SOCKET AND PLUG IT IN. BOTH THE 2716 AND THE ORIGINAL 9316 ROM USED BY APPLE ARE READ-ONLY-MEMORY DEVICES HOLDING 2K BY 8 BITS OF INFORMATION ("16K" ROMS), BUT THE PINOUT, OR ASSIGNMENT OF CHIP FUNCTIONS TO PIN NUMBERS IS SLIGHTLY DIFFERENT. TO USE THE 2716 IN A BOARD DESIGNED FOR A 9316, YOU NEED TO TIE PIN 21 TO 5 VOLTS (PIN 24) AND TIE PIN 18 TO GROUND (PIN 12). YOU COULD MODIFY THE PROM ITSELF, BUT YOU'RE LIABLE TO RUIN THE CHIP, AND IT CREATES A REAL MAGILLA IF YOU NEED TO REPROGRAM IT. (A ROM CARD, SUCH AS AN INTEGER CARD, CAN BE USED FOR 2716'S IF TWO JUMPERS ARE CONNECTED AT THE TOP OF THE CARD, AND ->ONLY<- 2716'S ARE USED IN ALL OF ITS SOCKETS AFTER THAT).

GET A 24-PIN, PREFERABLY LOW-PROFILE IC SOCKET, AND ORIENT IT WITH THE PINS UP AND THE NOTCH INDICATING THE 'PIN ONE' END TO THE RIGHT. IT SHOULD LOOK LIKE:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ !
!                              !
!                              !
!                             /
!                    (NOTCH)->!
!                             \
!                              !
! .  .     .  .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------
```

USING A LOW-WATTAGE SOLDERING IRON, SOLDER A SHORT PIECE OF 26-30 GAUGE WIRE BETWEEN PINS 21 AND 24, AND ANOTHER ONE BETWEEN PINS 12 AND 18. MAKE THE CONNECTION AS CLOSE TO THE SOCKET AS POSSIBLE, AND TRY TO AVOID GETTING ANY SOLDER ON THE ENDS OF PINS 12 AND 24. CUT OFF PINS 21 AND 18, AGAIN AS CLOSE AS POSSIBLE TO THE SOCKET. (PLUGGING ANOTHER SOCKET INTO THE ONE BEING MODIFIED WILL HELP TO PREVENT DISTORTION DURING THE SURGERY). THE SOCKET NOW LOOKS LIKE:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./  / ./ ./  / ./ ./ ./ !
!             X      X     /  !
!            /      /     /   !
!           /      /-------/   /
!    /---------/           !
!   /                       \
!  /                          !
! /  .      .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------

             X=NO PIN
```

DOUBLE CHECK THE CONNECTIONS ON THE BOTTOM OF THE SOCKET, AND PLUG THE 2716 INTO THE SOCKET, BEING CAREFUL TO MATCH THE NOTCHED END OF THE CHIP TO THE SOCKET. MAKE SURE THAT THE POWER TO THE APPLE IS TURNED OFF, AND PLUG THE ASSEMBLY INTO THE F8 SOCKET ON THE MOTHER BOARD WITH THE NOTCH TOWARD THE FRONT (KEYBOARD) END OF THE APPLE. CROSS YOUR FINGERS AND TURN ON THE APPLE. IF THERE IS NO FAMILIAR "BEEP", OR IF THE TV SCREEN STAYS WHITE, OR IF THE SYSTEM DOESN'T RESPOND TO THE RESET KEY, TURN OFF THE POWER AND EXAMINE THE CHIP AND SOCKET CAREFULLY TO FIND THE ERROR. IF BLACK CLOUDS OF SMOKE ROLL OUT FROM THE APPLE, FORGET WHERE YOU READ THIS. ACTUALLY, THE MOST COMMON MISTAKE OF INSERTING THE CHIP BACKWARDS IS SELDOM HARMFUL TO IT, BUT DOES LOCK UP THE APPLE'S BUS. REMEMBER THAT BOTH THE 2716 AND THE 9316 THAT YOU REMOVED CAN BE DAMAGED BY STATIC ELECTRICITY, SO HANDLE WITH CARE AND DON'T SCUFF YOUR FEET ON THE CAT.

```
================================================================================
DOCUMENT krckwczt.app
================================================================================

***************************************
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*      THE BASICS OF KRACKING I:      *
*                             *
*      ROMS AND PROMS AND  F8'S       *
*                             *
***************************************
```

ALONG WITH A NUMBER OF REQUESTS FOR MATERIAL USEFUL TO THOSE WHO ARE NOT YET IN THE RANKS OF PROFESSIONALS IN THIS FIELD, IT HAS BEEN POINTED OUT TO ME THAT I AM ALL TOO WILLING TO SUGGEST BURNING THIS PROM, INSTALLING THAT ROM, AND GENERALLY MAKING WHOLESALE HARDWARE CHANGES IN AN UNSUSPECTING APPLE, WITHOUT PROVIDING BACKGROUND INFORMATION FOR THE UP-AND-COMING KRACKISTS OF THE FUTURE.

THIS SERIES, WHILE AIMED AT THE BEGINNING TO INTERMEDIATE KRACKIST, WILL STILL ASSUME A REASONABLE KNOWLEDGE OF ASSEMBLY LANGUAGE. IF YOU FIND THESE DISCUSSIONS ARE STILL TOO HEAVY INTO MACHINE CODE FOR YOU, THEN IT'S BEST TO BUY A BOOK LIKE ROGER WAGNER'S "ASSEMBLY LINES" OR EQUIVALENT, AND STUDY IT CAREFULLY (IF, ON THE OTHER HAND, YOU FIND THAT THIS IS ALL BENEATH YOU, JUST KEEP A KNOWING SMIRK ON YOUR LIPS AS YOU SKIP LIGHTLY OVER THESE EPISODES - THERE MIGHT BE SOMETHING YOU MISSED BECAUSE YOU HAD A BAD HANGOVER ONE DAY IN KRACKING 101).

IN THIS AND FUTURE EPISODES IN THE 'BASICS OF KRACKING' SERIES, WE'LL DEAL WITH THE FUNDAMENTALS OF THE KRACKIST'S ART, STARTING WITH THE HOW (AND WHY) OF MAKING ALTERATIONS IN THE APPLE'S "PERMANENT" MEMORY. FIRST OF ALL, THE MOST IMPORTANT SINGLE TOOL AVAILABLE TO THE ASPIRING KRACKIST IS REPLACING THE AUTOSTART ROM ON THE MOTHER BOARD WITH AN "OLD MONITOR" ROM. WITH THIS ROM IN PLACE, YOU CAN HIT 'RESET' WHENEVER YOU WANT, AND ALWAYS BE RETURNED TO THE MONITOR FOR THE BEGINNING OF THE SNOOPING PROCESS. THIS CHANGE, INCIDENTALLY, WILL MAKE AVAILABLE TO YOU A REASONABLE SET OF "STEP AND TRACE" UTILITIES (SEE THE APPLE II REFERENCE MANUAL.   PP 51-53).

TO UNDERSTAND WHAT THE DIFFERENCES ARE BETWEEN THE TWO ROMS, LET'S TAKE A MINUTE TO EXAMINE WHAT PRESSING THE 'RESET' KEY DOES (OMIGOSH, MAUDE, THERE HE GOES AGAIN ON THAT DETAILED TECHNICAL CRAP!). INSTEAD OF GOING THROUGH THE KEYBOARD INPUT ROUTINE AT C000, THE RESET KEY IS CONNECTED DIRECTLY TO PIN 40 OF THE 6502 MICROPROCESSOR CHIP. WHEN THIS PIN IS CONNECTED TO GROUND (0 VOLTS), THE COMPUTER JUMPS UNCONDITIONALLY TO THE ADDRESS CONTAINED IN LOCATIONS FFFC AND FFFD. THIS IS NOT A TRUE INTERRUPT, SINCE THE APPLE FORGETS WHAT IT WAS DOING BEFORE THE LINE WAS "YANKED", BUT IT IS AN EXAMPLE OF 'VECTORING' OR SENDING THE COMPUTER TO A SPECIFIC PLACE BY SETTING AN ADDRESS INTO THE PROGRAM COUNTER. IN THE AUTOSTART ROM, THESE TWO LOCATIONS CONTAIN 62 FA, SO THE NEXT INSTRUCTION TO BE EXECUTED IS AT FA62.   THIS SERIES OF ROUTINES (SEE P. 143 AND PP. 36-38 OF THE REFERENCE MANUAL) CHECKS TO SEE IF THE COMPUTER IS BEING POWERED UP FOR THE FIRST TIME (COLDSTART) OR RESET WITH THE POWER ON (WARMSTART). IF IT IS A WARMSTART, THE SYSTEM JUMPS TO THE INSTRUCTIONS AT LOCATIONS 3F2 AND 3F3, AND BEGINS RUNNING THE PROGRAM FOUND THERE (USUALLY BASIC AT E000).

THE "OLD MONITOR" ROM, HOWEVER, HAS 59 FF STORED IN FFFC-D. THIS CAUSES AN

APPLE II (OR A II+ WITH AN INTEGER CARD AND THE RED SWITCH "UP") TO GO TO ROUTINES WHICH SET UP THE KEYBOARD FOR INPUT, THE TV FOR OUTPUT, AND WIND UP IN THE MONITOR WITH THE '*' PROMPT DISPLAYED. IN CONTRAST TO THE AUTOSTART ROM, WHERE ANYONE CAN TELL THE RESET BUTTON WHERE TO GO, THERE IS NO WAY TO PREVENT A RESET FROM GOING TO FF59 AND WINDING UP IN THE MONITOR. THIS IS OBVIOUSLY ESSENTIAL IF YOU WANT TO BREAK INTO A GAME AND START EXAMINING THE CODE, BUT IT HAS ITS OWN SET OF PROBLEMS.

IN THE PROCESS OF SETTING UP THE I/O DESCRIBED ABOVE, ESPECIALLY IN SETTING UP THE TEXT WINDOW ON THE SCREEN, A NUMBER OF LOCATIONS IN ZERO PAGE MUST BE CHANGED. THE FOLLOWING LOCATIONS WILL PROBABLY BE ALTERED (ALL HEX): 20,21,22,23,24,25,28,29,32,33,35, 36,37,38,39, AND 48.   WORSE THAN THAT, THE ENTIRE SCREEN SCROLLS UP ONE LINE WHEN THE MONITOR PROMPT IS PRINTED, WHICH LOSES THE ENTIRE TOP ROW OF THE TEXT SCREEN (LOCATIONS 400-427), AND ALTERS THE CONTENTS OF ALL THE OTHER LOCATIONS FROM 400-7FF, WITH THE EXCEPTION OF THE "SCRATCHPAD" REGIONS AT 478-47F, 4F8-4FF, ETC. (THE COMPUTER WIMP AT YOUR SCHOOL SAYS THAT THE TOP LINE "FALLS INTO THE BIT BUCKET", BUT YOU KNOW HOW EVERYONE FEELS ABOUT HIM.)

AS MOST SOFTWARE PROTECTORS KNOW, THIS WILL KEEP MOST OF THE AMATEURS OUT OF THE PROGRAM, AND YOU'LL SEE EVIDENCE OF THIS TECHNIQUE IN THE FORM OF A LOT OF "GARBAGE" ON THE TEXT SCREEN WHEN YOU RESET OUT OF A PROTECTED GAME. OUR JOB, THEN, IS TO KEEP THESE ZERO PAGE AND SCREEN MEMORY LOCATIONS FROM BEING LOST, SINCE MOST PROTECTION SCHEMES USE THESE AREAS IN SOME WAY OR OTHER (BR0DERBUND, FOR EXAMPLE, HAS RECENTLY BEEN STORING THE ADDRESS MARKER FOR THE DISK TRACK IN LOCATIONS 20, 21, AND 22).

THE SAFE WAY TO PREVENT INFORMATION FROM BEING LOST FROM THESE "VOLATILE" LOCATIONS IS TO TRANSFER ALL OF THE CONTENTS TO A SAFE AREA -- LOCATIONS 2000 & UP (OR 4000 & UP) WHERE A HI-RES PICTURE NORMALLY RESIDES. IN FACT, IT WOULD BE BEST TO SAVE EVERYTHING FROM 0 TO 8FF, SINCE BOOTING A DISKETTE TO SAVE THE DATA ALSO DESTROYS LOCATIONS 800-8FF. (REMEMBER THE FIRST LAW OF DISK KRACKING - TRACK 0, SECTOR 0 ALWAYS STARTS WITH D5 AA 96 AND ALWAYS LOADS INTO 800-8FF). BECAUSE THIS IS THE BEGINNING CLASS, LET'S LOOK AT TWO EXAMPLES OF SHORT BINARY SUBROUTINES THAT WILL DO THE "SAVE" FOR US. BOTH START, AS WILL BE EXPLAINED LATER, AT LOCATION FECD IN THE F8 ROM.   THE FIRST IS THE MOST STRAIGHTFORWARD AND EASIST TO FOLLOW:

```
  LDY  #$00    ;CLEAR Y-REGISTER
  LDA  $00,Y   ;GET A BYTE FROM 0+Y
  STA  $2000,Y ;STORE AT 2000+Y
  LDA  $0100,Y ;THEN FROM 100+Y
  STA  $2100,Y ;TO 2100+Y
  LDA  $0200,Y ;AND SO ON UNTIL
  STA  $2200,Y ;WE HAVE COVERED
  LDA  $0300,Y ;ALL THE MEMORY
  STA  $2300,Y ;'PAGES' FROM 0 TO 8
  LDA  $0400,Y ;AND STORED INTO
  STA  $2400,Y ;PAGES 20 TO 28
  LDA  $0500,Y
  STA  $2500,Y
  LDA  $0600,Y
  LDA  $2600,Y
  LDA  $0700,Y
  STA  $2700,Y
  LDA  $0800,Y
  STA  $2800,Y
  INY        ;THEN ADD 1 TO Y-REG
  BNE  $FED0   ;AND REPEAT IF < 256
  JMP  $FF59   ;WHEN WE'RE ALL DONE
             ;JUMP TO MONITOR START
```

THIS 61-BYTE ROUTINE, IF IT COULD BE EXECUTED AUTOMATICALLY WHEN THE RESET KEY IS PRESSED, WOULD SAFELY STASH ALL OF THE CHANGEABLE MEMORY AND EXIT GRACEFULLY INTO THE MONITOR.

A MORE COMPACT AND GENERAL, BUT LESS OBVIOUS ROUTINE IS SHOWN BELOW. IT IS INCLUDED BECAUSE IT IS TYPICAL OF THE "MEMORY MOVE PROGRAMS" THAT WE WILL EVENTUALLY HAVE TO WRITE IN KRACKING ALMOST ANY PROGRAM.

```
  LDY  #$00    ;CLEAR Y-REGISTER
  LDA  $00,Y   ;XFER THE ZERO PAGE TO
  STA  $2000,Y ;2000-20FF SO WE CAN USE
  INY        ;THE ZERO PAGE MEMORY
  BNE  $FED0   ;FOR THE OTHER MOVES
  LDA  #$00    ;SET UP LOCNS 0 & 1 AS A
  STA  $00     ;2-BYTE POINTER FOR THE
  STA  $02     ;SOURCE ADDRESS, USE 2&3
  LDA  #$01    ;AS 2-BYTE POINTER FOR
  STA  $01     ;THE DESTINATION ADDRESS
  LDA  #$21    ;STARTING AT $2100
  STA  $03
  LDA  ($00)<- ;GET A BYTE FROM 100-UP
  STA  ($02) ^ ;STORE AT 2100-UP
  INC  $02   ^ ;INCREMENT LO-ORDER BYTE
  INC  $00   ^ ;OF SOURCE & DESTINATION
  BNE  ->->->^ ;(BACK TO LDA ($00) IF
           ^ ;LO-ORDER IS <256
  INC  $03   ^ ;IF LO-ORDER=0, INC THE
  INC  $01   ^ ;HI BYTE OF EACH
  LDA  $01   ^ ;CHECK TO SEE IF HI-BYTE
  CMP  $#09  ^ ;IS 9 -WE'RE THRU AT 8FF
  BNE  ->->->^ ;IF NOT, LOOP BACK TO
             ;THE LOAD/STORE UNTIL
             ;WE'RE ALL DONE
  JMP  $FF59   ;EXIT THRU MONITOR
```

UNLIKE THE FIRST ROUTINE, THIS ONE (AT 47 BYTES) USES RAM LOCATIONS 0 THROUGH 3, SO THE ZERO PAGE MUST BE TRANSFERRED BEFORE IT IS ALTERED BY USING THOSE ADDRESSES AS POINTERS. WHILE THE FIRST ROUTINE MUST GROW BY SIX BYTES FOR EACH ADDITIONAL PAGE TRANSFERRED, THE SECOND NEEDS ONLY TO HAVE THE "9" IN THE COMPARE STATEMENT CHANGED TO THE APPROPRIATE VALUE ONE HIGHER THAN THE LAST PAGE NUMBER BEING TRANSFERRED.

TO RETURN TO THE BUSINESS OF ALTERING ROMS, IT IS EASY TO SEE THAT AN AUTOSTART ROM COULD BE MADE TO BEHAVE LIKE AN OLD ROM JUST BY CHANGING LOCATIONS FFFC-D TO 59 FF FROM 62 FA. (A NOTE TO THE FAINT-HEARTED--YOU CAN BUY AN OLD MONITOR F8 ROM FOR ABOUT $10 AND PLUG IT DIRECTLY INTO YOU APPLE'S F8 SOCKET, BUT YOU WON'T HAVE ALL THE BENEFITS WE'VE BEEN TALKING ABOUT). AS LONG AS WE'RE GOING TO THE EFFORT OF MAKING A CHANGE, THOUGH, WE MIGHT AS WELL ADD ONE OF THE ROUTINES ABOVE AND ALLOW THE NEW ROM TO SAVE THE VOLATILE MEMORY FOR US. TO DO THIS, WE'LL HAVE TO GIVE UP SOMETHING IN THE ROM, AND THE MOST EASILY SURRENDERED AREA FOR MOST OF US IS THE TAPE READ/SAVE ROUTINES AT $FECD. IF WE THEN CHANGED FFFC-D TO CD FE, THE MEMORY FROM 0 TO 8FF WOULD BE SAVED TO 2000-28FF EVERY TIME THE 'RESET' KEY WAS PRESSED. SINCE IT'S SOMETIMES INCONVENIENT TO HAVE THAT HAPPEN WHEN THE RESET KEY IS PRESSED, WE CAN REQUIRE

THAT A SPECIFIC KEY BE ALSO PRESSED TO MAKE IT OCCUR. THESE FEW INSTRUCTIONS INSERTED BEFORE EITHER OF THE ROUTINES ABOVE WILL GIVE A "RESET AND SAVE" WHEN THE "-" KEY IS HELD DOWN (OR WAS THE LAST KEY PRESSED), WHILE GIVING A REGULAR "OLD RESET" THE REST OF THE TIME.

```
  LDA  $C000  ;LOOK AT THE KEYBOARD
  ROL        ;MASK OFF HIGH BIT
  CMP  #$5A   ;WAS IT "-"?($2D X 2=$5A)
  BNE  ->->-> ;IF NOT, BRANCH TO THE
          ! ;LOCATION WITH THE
          ! ;"JUMP FF59" INSTRUCTION
          ! ;AT THE END OF THE SAVE
          ! ;SUBROUTINE.
```

OK, OK - WE ALL AGREE THAT THESE WOULD BE NEAT THINGS TO HAVE IN THE F8 ROM, SO HOW DO WE GET IT THERE? FIRST, GET HOLD OF A PROMBURNER (PROMBLASTER, EPROM PROGRAMMER, ETC.) THAT WILL PROGRAM 2716 EPROMS. EACH ONE IS DIFFERENT, SO I WON'T TRY TO GIVE DETAILED INSTRUCTIONS ON THE ACTUAL PROGRAMMING. BUY OR BORROW A FRIEND'S OLD F8 ROM (OR GET THE BINARY FILE) THEN TYPE IN OR LOAD IN THE CHANGES YOU WANT TO MAKE AT FECD & UP AND AT FFFC-D, AND PROGRAM A 2716 EPROM WITH OUR MODIFIED VERSION OF APPLE'S F8 MONITOR ROM.

ALL THAT REMAINS TO TAKE FULL ADVANTAGE OF THE NEW F8 ROM IS TO MAKE A SLIGHTLY MODIFIED SOCKET AND PLUG IT IN. BOTH THE 2716 AND THE ORIGINAL 9316 ROM USED BY APPLE ARE READ-ONLY-MEMORY DEVICES HOLDING 2K BY 8 BITS OF INFORMATION ("16K" ROMS), BUT THE PINOUT, OR ASSIGNMENT OF CHIP FUNCTIONS TO PIN NUMBERS IS SLIGHTLY DIFFERENT.   TO USE THE 2716 IN A BOARD DESIGNED FOR A 9316,
YOU NEED TO TIE PIN 21 TO 5 VOLTS (PIN 24) AND TIE PIN 18 TO GROUND (PIN 12). YOU COULD MODIFY THE PROM ITSELF, BUT YOU'RE LIABLE TO RUIN THE CHIP, AND IT CREATES A REAL MAGILLA IF YOU NEED TO REPROGRAM IT. (A ROM CARD, SUCH AS AN INTEGER CARD, CAN BE USED FOR 2716'S IF TWO JUMPERS ARE CONNECTED AT THE TOP OF THE CARD, AND ->ONLY<- 2716'S ARE USED IN ALL OF ITS SOCKETS AFTER THAT).

GET A 24-PIN, PREFERABLY LOW-PROFILE IC SOCKET, AND ORIENT IT WITH THE PINS UP AND THE NOTCH INDICATING THE 'PIN ONE' END TO THE RIGHT. IT SHOULD LOOK LIKE:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ ./ !
!                             !
!                             !
!                            /
!                    (NOTCH)->!
!                            \
!                             !
! .  .      .  .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------
```

USING A LOW-WATTAGE SOLDERING IRON, SOLDER A SHORT PIECE OF 26-30 GAUGE WIRE BETWEEN PINS 21 AND 24, AND ANOTHER ONE BETWEEN PINS 12 AND 18. MAKE THE CONNECTION AS CLOSE TO THE SOCKET AS POSSIBLE, AND TRY TO AVOID GETTING ANY SOLDER ON THE ENDS OF PINS 12 AND 24. CUT OFF PINS 21 AND 18, AGAIN AS CLOSE AS POSSIBLE TO THE SOCKET. (PLUGGING ANOTHER SOCKET INTO THE ONE BEING MODIFIED WILL HELP TO PREVENT DISTORTION DURING THE SURGERY). THE SOCKET NOW LOOKS LIKE:

```
---------------------------------------
!  13 14 15 16 17 18 19 20 21 22 23 24!
! ./ ./ ./ ./ ./  / ./ ./  / ./ ./ ./ !
!             X      X     /  !
!            /      /     /   !
!           /      /-------/   /
!    /---------/            !
!   /                        \
!  /                          !
! /  .      .  .  .  .  .  .  .  .  .  .  !
!/  /  /  /  /    /  /  /  /  /  /  /   !
!12 11 10 9  8    7  6  5  4  3  2  1   !
---------------------------------------

             X=NO PIN
```

DOUBLE CHECK THE CONNECTIONS ON THE BOTTOM OF THE SOCKET, AND PLUG THE 2716 INTO THE SOCKET, BEING CAREFUL TO MATCH THE NOTCHED END OF THE CHIP TO THE SOCKET. MAKE SURE THAT THE POWER TO THE APPLE IS TURNED OFF, AND PLUG THE ASSEMBLY INTO THE F8 SOCKET ON THE MOTHER BOARD WITH THE NOTCH TOWARD THE FRONT (KEYBOARD) END OF THE APPLE. CROSS YOUR FINGERS AND TURN ON THE APPLE. IF THERE IS NO FAMILIAR "BEEP", OR IF THE TV SCREEN STAYS WHITE, OR IF THE SYSTEM DOESN'T RESPOND TO THE RESET KEY, TURN OFF THE POWER AND EXAMINE THE CHIP AND SOCKET CAREFULLY TO FIND THE ERROR. IF BLACK CLOUDS OF SMOKE ROLL OUT FROM THE APPLE, FORGET WHERE YOU READ THIS. ACTUALLY, THE MOST COMMON MISTAKE OF INSERTING THE CHIP BACKWARDS IS SELDOM HARMFUL TO IT, BUT DOES LOCK UP THE APPLE'S BUS. REMEMBER THAT BOTH THE 2716 AND THE 9316 THAT YOU REMOVED CAN BE DAMAGED BY STATIC ELECTRICITY, SO HANDLE WITH CARE AND DON'T SCUFF YOUR FEET ON THE CAT.

```
***************************************
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*      THE BASICS OF KRACKING II      *
*                             *
*     SINGLE-LOAD GAMES, STARTING     *
*     LOCATIONS, AND OBFUSCATION.     *
*                             *
***************************************
```

THE FIRST IN THIS SERIES WAS STRAIGHTFORWARD, SINCE THE HARDWARE RESET IS A NECESSITY TO BEGIN KRACKING. AFTER THAT, THE PATH DIVIDES, AND THERE ARE MANY MANY WAYS TO PRODUCING AN UNPROTECTED VERSION OF A PROGRAM. THE PATH YOU FOLLOW IS GOVERNED BY THREE THINGS: THE KIND OF PROGRAM, THE TYPE OF PROTECTION EMPLOYED, AND YOUR OWN PERSONAL STYLE (STYLE, BY THE WAY, IS PRIMARILY THE RESULT OF LIMITATIONS. TRY TO KEEP AN OPEN MIND AND DEVELOP AS MUCH VERSATILITY AS POSSIBLE). THE EASIEST KIND OF PROGRAM TO DEAL WITH IS THE ONE THAT IS SEEN LESS FREQUENTLY EVERY MONTH: THE "SINGLE-LOAD" PROGRAM OR GAME. THESE ARE PROGRAMS WHICH ARE LOADED IN FROM DISK ONLY ONCE, AND THEN ARE RUN STRICTLY FROM MEMORY WITH NO DISK ACCESS. IN THE GOOD OLD DAYS, ALMOST EVERY GAME WAS LIKE THIS, AND REMOVING PROTECTION WAS NOT THAT DIFFICULT. ON THE OTHER HAND, WHEN YOU READ SOMETHING LIKE OLAF LUBECK'S CHALLENGE IN TRACK 17, SECTOR D OF CANNONBALL BLITZ: "YOU'LL NEVER CRACK IT", THERE'S MORE SATISFACTION WHEN YOU GET TO SAY "OH, YES I DID!".

IN ORDER TO BECOME PROFICIENT AT THIS AND THE TECHNIQUES TO BE DISCUSSED IN FUTURE EPISODES, YOU WILL HAVE TO GET USED TO COMMITTING A VERY UNNATURAL ACT: INTERPRETING ASSEMBLER CODE WITH NO COMMENTS OR INSTRUCTIONS TO GUIDE YOU. THE DISASSEMBLER (MONITOR 'L' COMMAND) IS A GREAT HELP IN THIS WORK, SINCE IT TRANSLATES MACHINE CODE INTO ASSEMBLER MNENONICS, BUT THE REAL BURDEN FALLS ON THE INGENUITY OF THE KRACKIST. THERE IS NO SUBSTITUTE FOR EXPERIENCE, AND NO ONE CAN TEACH YOU HOW TO DO IT BEYOND POINTING OUT SOME OF THE TECHNIQUES WE USE, AND WARNING YOU ABOUT SOME OF THE TRICKS USED TO KEEP YOU FROM SUCCEEDING.

THE PHILOSOPHY OF ATTACK WITH THESE GAMES IS TO FIND THE STARTING LOCATION--THE ADDRESS WHICH WILL ALWAYS RESTART THE GAME, AND THEN TO SAVE THE GAME (PROGRAM) AS A NORMAL DOS 3.3 BINARY FILE. AS A SIMPLE EXAMPLE OF A STARTING LOCATION, YOU PROBABLY ALREADY KNOW THAT WHEN YOU MESS UP WITH APPLE'S "FID" PROGRAM, YOU CAN RESTART BY TYPING '803G' FROM THE MONITOR. AT ONE TIME, BEFORE THE PUBLISHERS GOT SMART, A STARTING LOCATION WAS LIKELY TO BE A COMMON, EVEN NUMBER LIKE $800, C00, 4000, OR 6000, AND IT'S STILL WORTH CHECKING THESE 'OLD FAVORITES' IN CASE YOU FIND A NAIVE OR LAZY AUTHOR. IF THESE FAIL, WE WILL HAVE TO BEGIN THE PROCESS OF MEMORY SNOOPING. THIS IS THE INTRODUCTION TO THE UNGLAMOROUS ACTIVITY THAT OCCUPIES MOST OF THE TIME OF THE DEDICATED KRACKIST. AS ALWAYS, INSPECTOR AND WATSON IN ROM ARE HIGHLY RECOMMENDED, SINCE THEY MAKE THE PROCESS INFINITELY EASIER. WHAT WE ARE TRYING TO DO IS DIRECTLY LOCATE THE
BEGINNING ADDRESS OF THE PROGRAM, OR TO SEARCH BACK TO IT FROM SOMETHING WE CAN RECOGNIZE.

SINCE MANY GAMES BEGIN BY DISPLAYING A HI-RES "BANNER" OR GAME SCREEN, A GOOD PLACE TO START LOOKING IS THE SERIES OF INSTRUCTIONS THAT SET UP THE HI-RES SCREEN (THERE IS A DISCUSSION OF THIS IN THE DOC FOR MASTERKEY PLUS, BUT THEY MAKE A FEW TOO MANY ASSUMPTIONS). APPLE'S SCREEN DISPLAY, AS YOU PROBABLY KNOW, IS SET UP BY ACCESSING SOME "SOFT SWITCHES". IN HEX, THESE ARE LOCATIONS $C050 TO C057 (SORRY, BUT IF YOU'RE GOING TO LEARN THE GENTLE ART OF KRACKING, YOU'LL HAVE TO BECOME FLUENT IN HEXADECIMAL--WE WON'T PULL ANY PUNCHES WHEN IT COMES TO NUMBER SYSTEMS). IT DOESN'T MATTER WHAT YOU DO TO THESE LOCATIONS, AS LONG AS YOU MAKE A REFERENCE, SO THE FOLLOWING INSTRUCTIONS ALL ESTABLISH GRAPHICS MODE:

```
   LDA $C050,     BIT $C050,   ROL $C050
   STA $C050,     CMP $C050,   EOR $C050
```

(ALSO, THIS ONE: LDY #$71; AND $BFAF,Y)

MANY AUTHORS HAVE ESTABLISHED THE HABIT, HOWEVER, OF WRITING THE SEQUENCE

```
   LDA $C054   (SELECT PRIMARY PAGE)
   LDA $C057   (SELECT HI-RES GRAPHICS)
   LDA $C050   (SELECT GRAPHICS MODE)
```

AND SOMETIMES,

```
   LDA $C052   (PURE GRAPHICS SCREEN).
```

TO FIND THESE INSTRUCTIONS, USE THE INSPECTOR'S 'FIND' FUNCTION, AND PROGRAM IT TO SEARCH FOR THE TWO-BYTE SEQUENCES OF '50 C0' AND '57 C0'. GENERALLY, AS LONG AS THE WRITERS AREN'T DELIBERATELY TRYING TO CONFUSE YOU, YOU WILL FIND ONE TO SEVERAL LOCATIONS WHERE THESE SEQUENCES ARE CLOSE TO EACH OTHER. YOU WILL ALSO FIND SOME ADDRESSES THAT DON'T REALLY CONTAIN A SCREEN REFERENCE, SINCE THE SEARCH IS ONLY FOR TWO BYTES (FOR YOU TRIVIA/ STATISTICS BUFFS OUT THERE, A GIVEN TWO-BYTE SEQUENCE WOULD OCCUR LESS THAT ONCE IN THE ENTIRE RAM MEMORY SPACE FROM 0 TO $BFFF IF THE DISTRIBUTION WERE TRULY RANDOM. IT'S NOT.).

TO SEE IF EACH OCCURANCE OF THE PATTERN IS THE STARTING LOCATION, LOOK BACKWARDS UNTIL YOU FIND AN ABSOLUTE END FOR THE PREVIOUS SUBROUTINE SUCH AS 'RTS' OR 'JMP'. YOUR SUBROUTINE SHOULD BEGIN IMMEDIATELY AFTER THAT, AND YOU SHOULD ASSUME FOR THE MOMENT THAT IT'S THE STARTING LOCATION. IF, FOR EXAMPLE, THE LOCATION YOU FOUND IS $4123, TEST IT BY RELOADING THE GAME, RESETTING IT, AND TYPING '4123G'. IF IT RUNS, SIT BACK AND GLOAT, OTHERWISE READ ON (IT SOUNDS UNNECESSARY TO RELOAD, BUT THE INSPECTOR USES A FEW LOCATIONS IN PAGES 0, 2, AND 3, SO IT'S BEST TO BE SAFE). IF MURPHY'S LAW OF DYNAMIC NEGATIVES IS WITH YOU AND THE GAME DIDN'T START, IT'S USUALLY BECAUSE YOU HAVEN'T FOUND THE TRUE STARTING LOCATION. YOU THEN NEED TO TRACE BACK FURTHER IN THE PROGRAM SEQUENCE TO FIND THE REAL START.

THERE ARE THREE WAYS FOR ANOTHER ROUTINE TO GET TO THE ONE YOU'RE LOOKING AT: JMP, JSR, AND THE FAMILY OF BRANCH INSTRUCTIONS. TO ELIMINATE THE THIRD POSSIBILITY, KEEP IN MIND THAT BRANCHES CAN REACH UP TO $7F (127) LOCATIONS AWAY FROM EITHER DIRECTION. THIS IS EQUAL TO ABOUT 60 INSTRUCTIONS, SO YOU SHOULD REVIEW ABOUT ONE FULL PAGE OF DISASSEMBLY PRINTOUT (THREE SCREENSFUL) BEFORE AND RARELY AFTER WHAT LOOKED LIKE A POSSIBLE START. IF YOU FIND A 'BNE 4123', OR 'BCC 4123', ETC., YOU WILL HAVE TO TRACK BACK TO THE BEGINNING OF THAT ROUTINE AND TRY AGAIN. REPEAT THIS PROCESS UNTIL YOU FIND A LOCATION THAT CAN ONLY BE REACHED BY A JMP OR JSR.

TO FIND OUT HOW THE PROGRAM GOT TO THIS LOCATION, DO A 3-BYTE SEARCH WITH THE INSPECTOR FOR A JSR $4123: 20 23 41. IF NOTHING SHOWS UP, TRY THE JMP $4123: 4C 23 41. ONE OF THESE MUST PRODUCE A REFERENCE, OR YOU MESSED UP THE EARLIER CHECK FOR BRANCHES. ONCE YOU FIND THE EARLIER REFERENCE, GO THROUGH THE SAME PROCEDURE TO FIND THE START OF THIS ROUTINE, AND TRY IT OUT AS A STARTING LOCATION FOR THE GAME. IF IT DOESN'T WORK, TRY ONE MORE STEP FURTHER BACK (KRAKOWICZ'S FOURTH LAW OF KRACKING SAYS THAT IF YOU HAVE TO GO BACK MORE THAN TWO STEPS, YOU'RE PROBABLY NOT ON THE RIGHT TRAIL).

A NUMBER OF GAMES STILL DO US THE FAVOR OF PUTTING UP A SCREEN, PERHAPS PLAYING A LITTLE MUSIC, AND THEN WAITING FOR THE SPACE BAR OR OTHER KEY TO BE PRESSED. IF IT'S NOT POSSIBLE TO FIND THE SCREEN SETUP, WE STILL HAVE A FAIRLY OBVIOUS "HOOK" INTO FINDING THE STARTING ADDRESS, AND IN MANY CASES THE GAME CAN BE SAVED 'AS IS' BY USING THE KEYBOARD ROUTINE AS THE STARTING ADDRESS. DON'T WORRY FOR NOW ABOUT EXACTLY HOW WE WILL "SAVE THE GAME". WE'LL GO THROUGH THAT CAREFULLY AND THOROUGHLY IN THE NEXT EPISODE.

SINCE THE KEYBOARD ADDRESS IS C000, WE CAN USUALLY LOCATE ALL THE INPUTS BY SEARCHING FOR THE 3-BYTE SEQUENCE OF 'AD 00 C0' WITH THE INSPECTOR. OCCASIONALLY, THE X OR Y REGISTER IS USED TO LOAD KEYBOARD DATA, SO THE SEQUENCES AC 00 C0 AND AE 00 C0 SHOULD BE TRIED IF THE FIRST COMES UP BLANK (ONLY THE REAL BASTARDS LIKE SIRIUS USE LDY #$67; LDA $BF99,Y FOR THE KEYBOARD INPUT). ALSO, KEEP IN MIND THAT ALL THE ADDRESSES FROM C000 TO C00F WILL ACCESS THE KEYBOARD, AND IF SOMEONE WAS REALLY DETERMINED TO CONFUSE YOU THEY COULD USE C007 ONE TIME, C00D THE NEXT, AND SO ON. IF YOU KNOW THAT THE GAME USES THE KEYBOARD AND THE PRELIMINARY SEARCHES DON'T SHOW HOW, KEEP ON LOOKING FOR THESE ADDRESSES, OR THE SIRIUS-TYPE COMPUTED ADDRESSES. IT PROBABLY MEANS THEY HAVE SOMETHING TO KIDE, AND LOCATING THE KEYBOARD READ WILL REVEAL ENOUGH TO MAKE THE SEARCH WORTHWHILE.

IF THE PROGRAM IS WAITING FOR THE SPACE BAR, YOU WILL USUALLY FIND A SEQUENCE LIKE:

```
  78E0: LDA $C000   ;READ THE KEYBORARD
        BPL $78E0   ;NO KEY PRESSED
```

```
      STA $C010   ;RESET KBD STROBE
       *CMP #$A0    ;WAS IT SPACE?
       *BNE $78E0   ;NOPE, KEEP TRYING
      JMP $6012   ;YES, GO TO START
```

*THESE TWO LINES ARE ELIMINATED IF PRESSING ANY KEY WILL START THE GAME.

TO CHECK OUT 6012 AS A STARTING ADDRESS, SET UP TO VIEW THE HI-RES SCREEN (OTHERWISE THE GAME MIGHT BE RUNNING WHILE YOU WATCH A BLANK TEXT SCREEN) WITH: C050 (CR) C057 (CR), THEN TYPE 6012G. AS BEFORE, YOU WILL KNOW AT ONCE IF YOU WERE SUCCESSFUL.

ANOTHER WAY TO FIND A RESTART POINT IS TO SEARCH THROUGH THE KEYBOARD INPUT ROUTINES FOR A RESTART KEY. IT HAS BECOME CONVENTIONAL TO USE CTRL-R AS THE RESTART COMMAND (OCCASIONALLY CTRL-S OR CTRL-B), AND THIS IS EVEN EASIER TO TRACE. IN ONE OF THE ROUTINES FOLLOWING A C000 REFERENCE, YOU WILL FIND A CMP #$92 (SEE THE REFERENCE MANUAL, P. 7 FOR THE HEX VALUES OF THE KEYBOARD). THE LOCATION BRANCHED TO OR JUMPED TO BY A SUCCESSFUL COMPARE WILL BE THE RESTART FOR THE GAME. AGAIN, YOU CAN SAVE THE GAME AS IS AND USE YOUR NEW-FOUND STARTING LOCATION.

IF THESE RELATIVELY SIMPLE APPROACHES FAIL, YOU'LL HAVE TO RESORT TO THE REAL GRUNT TYPE OF DETECTIVE WORK--LOOKING FOR SOMETHING PROMISING (WE'LL DISCUSS BOOT-TRACING AS AN ALTERNATIVE WAY OF GETTING TO THIS POINT IN ANOTHER EPISODE DEVOTED ENTIRELY TO THAT TECHNIQUE). LIKELY THINGS TO LOOK FOR ARE "SETUPS", WHERE A LOT OF ZERO PAGE LOCATIONS ARE INITIALIZED TO BEGIN THE GAME:

```
         LDA #$00
         STA $23
         STA $57
         LDA #$12
         STA $30
         LDA #$E9
         STA $72
         ETC.
         ETC
```

OR, SOMETIMES, A GAME START IS INDICATED BY A SUBROUTINE SEQUENCE WHICH MAPS OUT THE PATH FOR THE GAME (THIS IS AN INDICATION OF AN EXPERIENCED, WELL-DISCIPLIED PROGRAMMER, AND THUS IS MORE COMMONLY SEEN IN BUSINESS OR PROFESSIONAL PROGRAMS; RARELY IN GAME PROGRAMMING).

```
      JSR $8CD
      JSR $CE4
      JSR $2020
      JSR $203D
      JSR $8FE
      ETC.
```

AND, ALTHOUGH IT'S LESS OFTEN THE START OF A PROGRAM OR GAME, A "JUMP TABLE" CAN BE A SIGNIFICANT CLUE TO THE ORGANIZATION OF THE PROGRAM:

```
      JMP $204D
      JMP $2433
      JMP $EF2
      JMP $2077
      ETC.
```

UNFORTUNATELY, SNOOPING FOR THESE IS A TIME-CONSUMING, HIT-AND-MISS OPERATION - THE REAL STARTING ADDRESS CAN BE ANYWHERE FROM 0000 TO BFFF (OR EVEN VIA A BASIC SUBROUTINE IN D000-F7FF, BUT I DON'T WANT TO DISCOURAGE YOU YET).

WHILE IT WILL BE DISCONCERTING TO THE BEGINNER, AS YOU GET MORE EXPERIENCE YOU BEGIN TO ENJOY DEFEATING VARIOUS DELIBERATE ATTEMPTS TO THROW YOU OFF THE TRAIL--THE GENERAL SUBJECT OF OBFUSCATION, OR INTENTIONAL LACK OF CLARITY. BECAUSE THE MAJOR SOFTWARE COMPANIES KNOW WE'RE OUT HERE WAITING FOR THEIR LATEST OUTPUT, THEY OFTEN TRY TO MISDIRECT US OR FIND INNOVATIVE WAYS OF HIDING SENSITIVE PORTIONS OF THE PROGRAM WITH A VARIETY OF TECHNIQUES. TAKE A LOOK AT THE FOLLOWING PIECE OF CODE FROM ON-LINE'S CANNONBALL BLITZ:

```
59E4- CE E7 59    DEC   $59E7
59E7- CF        ???
59E8- EA        NOP
59E9- 59 EF EA    EOR   $EAEF,Y
59EC- 59 AD 51    EOR   $51AD,Y
59EF- C0 AD     CPY   #$AD
59F1- 54        ???
59F2- C0 AD     CPY   #$AD
59F4- 57        ???
59F5- C0 AD     CPY   #$AD
59F7- 52        ???
59F8- C0 20     CPY   #$20
59FA- 60        RTS
59FB- 5B        ???
59FC- 20 C5 5B    JSR   $5BC5
59FF- 20 4E 5B    JSR   $5B4E
```

THIS IS AN EXAMPLE OF "SELF-MODIFYING CODE"-INSTRUCTIONS THAT CHANGE AS THE PROGRAM IS RUN. IT'S DANGEROUS AND GENERALLY POOR PROGRAMMING PRACTICE, BUT IT CAN BE USED TO THROW THE DOGS OFF THE SCENT. AT FIRST GLANCE, IT LOOKS LIKE DATA OR GARBAGE STUCK IN BEFORE SOME REAL CODE. LET'S LOOK AT EXACTLY HOW IT WORKS.   EXECUTING THE FIRST INSTRUCTION CHANGES THE SECOND INSTRUCTION FROM JUNK
INTO A LEGAL INSTRUCTION:

```
59E4- CE E7 59    DEC   $59E7
59E7- CE EA 59    DEC   $59EA
59EA- EF        ???
59EB- EA        NOP
59EC- 59 AD 51    EOR   $51AD,Y
59EF- C0 AD     CPY   #$AD
```

(IF YOU HAVE AN OLD MONITOR ROM, YOU CAN TYPE 59E4S TO EXECUTE THE FIRST INSTRUCTION). IF WE EXECUTE THE SECOND INSTRUCTION, THE ENTIRE PICTURE CHANGES:

```
59E4- CE E7 59    DEC   $59E7
59E7- CE EA 59    DEC   $59EA
59EA- EE EA 59    INC   $59EA
59ED- AD 51 C0    LDA   $C051
59F0- AD 54 C0    LDA   $C054
59F3- AD 57 C0    LDA   $C057
59F6- AD 52 C0    LDA   $C052
59F9- 20 60 5B    JSR   $5B60
59FC- 20 C5 5B    JSR   $5BC5
59FF- 20 4E 5B    JSR   $5B4E
5A02- A9 04     LDA   #$04
```

```
5A04- 8D EC B7    STA   $B7EC
5A07- A9 00     LDA   #$00
5A09- 8D EB B7    STA   $B7EB
5A0C- A9 00     LDA   #$00
5A0E- 8D F0 B7    STA   $B7F0
5A11- A9 60     LDA   #$60
5A13- 8D F1 B7    STA   $B7F1
5A16- A9 40     LDA   #$40
5A18- 20 45 5A    JSR   $5A45
5A1B- 10 01     BPL   $5A1E
5A1D- A9 20     LDA   #$20
5A1F- 91 5A     STA   ($5A),Y
5A21- AD 50 C0    LDA   $C050
5A24- A9 09     LDA   #$09
```

  SUDDENLY, THE SCREEN SETUP CODE THAT WAS ALWAYS THERE POPS INTO VIEW. THIS POINTS OUT THE VALUE OF SEARCHING WITH THE INSPECTOR, SINCE EVEN THE CLOSEST SCRUTINY WOULD PROBABLY NOT HAVE MADE YOU SUSPECT WHAT WAS ACTUALLY HERE. NOTICE, TOO, THAT THE THIRD INSTRUCTION INCREMENTS 59EA, SO ONCE IT'S BEEN RUN, IT'S OBSCURED AGAIN.

  ANOTHER STANDARD TRICK, ALSO SHOWN IN THIS EXAMPLE, IS CALLED "FALSE DISASSEMBLY", AND IS DEAR TO EDU-WARE, ON-LINE, IDSI, AND SCIENTIFIC RESEARCH ASSOCIATES. HERE, EXTRA BYTES ARE ADDED FOR THE SOLE PURPOSE OF GIVING A FALSE INDICATION OF PROGRAM FLOW; THE FAKE BYTES ARE THEN BRANCHED AROUND. LOOK CLOSELY AT THE INSTRUCTION IN 5A1B-IT SAYS BPL 5A1E. THE NEXT INSTRUCTIONS IN SEQUENCE APPEAR TO THE CASUAL EYE TO BE LDA $#20; STA ($5A),Y.   ACTUALLY, THE NEXT INSTRUCTION IS JSR $5A91.   THIS IS CRUCIAL, SINCE THIS SUBROUTINE LOADS IN
THE GAME AND DOES A NIBBLE COUNT. TO SEE A WHOLE BUNCH OF FALSE DISASSEMBLIES IN A ROW, LOOK AT THE CODE IN THE ACTUAL SUBROUTINE:

```
5A91- A9 00     LDA   #$00
5A93- 10 01     BPL   $5A96
5A95- 20 A8 59    JSR   $59A8
5A98- 00        BRK
5A99- 27        ???
5A9A- C8        INY
5A9B- D0 FA     BNE   $5A97
5A9D- 85 10     STA   $10
5A9F- F0 01     BEQ   $5AA2
5AA1- A9 A9     LDA   #$A9
5AA3- 20 59 00    JSR   $0059
5AA6- 27        ???
5AA7- C8        INY
5AA8- C8        INY
5AA9- D0 F9     BNE   $5AA4
5AAB- 85 11     STA   $11
5AAD- 49 B7     EOR   #$B7
5AAF- 48        PHA
5AB0- A5 10     LDA   $10
5AB2- 49 11     EOR   #$11
5AB4- 48        PHA
5AB5- D0 01     BNE   $5AB8
5AB7- 4C 60 08    JMP   $0860
5ABA- 60        RTS
```

  I STRONGLY URGE YOU TO SIT DOWN AND FIGURE OUT EXACTLY WHAT THE REAL PROGRAM

IS HERE, AND IF POSSIBLE, WHAT IT DOES. COVER UP THE EXPLANATION BELOW, AND GO THROUGH THE CODE BYTE BY BYTE TO ELIMINATE THE FAKE BYTES. IT'S NOT JUST CHARACTER-BUILDING--IF YOU GO THROUGH A FEW OF THESE, YOU'LL LEARN TO RECOGNIZE THEM WHEN THEY POP UP.

THOSE OF YOU WHO REALLY WENT THROUGH IT, GIVE YOURSELVES FOUR KRACKING HONOR POINTS. FOR THE REST OF YOU, HERE'S A LISTING OF THE FUNCTIONAL EQUIVALENT (SOME ADDRESSES ARE CHANGED BECAUSE THE JUNK BYTES HAVE BEEN TAKEN OUT):

```
5A91- A9 00     LDA   #$00
5A93- A8        TAY
5A94- 59 00 27    EOR   $2700,Y
5A97- C8        INY
5A98- D0 FA     BNE   $5A94
5A9A- 85 10     STA   $10
5A9C- A9 20     LDA   #$20
5A9E- 59 00 27    EOR   $2700,Y
5AA1- C8        INY
5AA2- C8        INY
5AA3- D0 F9     BNE   $5A9E
5AA5- 85 11     STA   $11
5AA7- 45 B7     EOR   $B7
5AA9- 48        PHA
5AAA- A5 10     LDA   $10
5AAC- 49 11     EOR   #$11
5AAE- 48        PHA
5AAF- 60        RTS
```

THIS IS ALSO VALUABLE BECAUSE IT INTRODUCES THE CONCEPT OF "JUMPING THROUGH THE STACK". THE RTS INSTRUCTION TRANSFERS THE TWO BYTES ABOVE THE STACK POINTER IN PAGE ONE TO THE PROGRAM COUNTER, INCREMENTS THE LOW BYTE BY ONE, AND JUMPS TO THAT LOCATION. ORDINARILY, THE BYTES ON THE STACK WERE PLACED THERE AS A RETURN ADDRESS BY THE JSR INSTRUCTION. IN THIS CASE, IN VERY ROUNDABOUT FASHION, THE ON-LINERS HAVE PUSHED TWO BYTES ON THE STACK AND EXECUTED AN RTS, WHICH JUMPS TO THE LOCATION ONE HIGHER THAT THE VALUES STORED. THE STORY OF THE SUBROUTINE GOES LIKE THIS: CREATE A CHECKSUM BY EXCLUSIVE-ORING TOGETHER ALL THE BYTES FROM 2700 TO 27FF, AND STORE IT IN $10. THIS ALLOWS A CHECK TO SEE IF ANY OF THE BYTES IN THE NIBBLE COUNT ROUTINE WERE ALTERED. DO A SECOND CHECKSUM ON EVERY OTHER BYTE FROM 2700 TO 27FF, STARTING WITH A VALUE OF #$20. STORE THIS IN $11, THEN EXCLUSIVE-OR IT WITH #$B7 TO PRODUCE THE LOW BYTE OF THE RETURN ADDRESS:FF. PUSH THIS ON THE STACK, EXCLUSIVE-OR THE FIRST CHECKSUM WITH #$11 TO PRODUCE THE RETURN HIGH BYTE OF $26, THEN DO THE RTS TO JUMP TO 2700. WHEN YOU LOOK AT 2700, YOU FIND THIS:

```
2700- CE 03 27    DEC   $2703
2703- EF        ???
2704- 03        ???
2705- 27        ???
2706- AD 24 27    LDA   $2724
2709- 49 8A     EOR   #$8A
270B- D0 01     BNE   $270E
270D- 20 8D 24    JSR   $248D
2710- 27        ???
2711- D0 01     BNE   $2714
2713- 4C A0 25    JMP   $25A0
2716- 98        TYA
2717- 59 00 27    EOR   $2700,Y
271A- 99 00 27    STA   $2700,Y
```

```
271D- C8        INY
271E- D0 F6     BNE   $2716
```

(YOU SEE, NOW THAT WE'RE FAMILIAR WITH THIS KIND OF TRICK, THERE'S NOTHING TO DECODING THAT MESS, IS THERE?)

STAY TUNED FOR NEXT WEEK, WHEN WE FINISH THIS SUBJECT BY ANSWERING THE BURNING QUESTION "WHAT IS THE WINDOW-SHADE TECHNIQUE?", AND PROCEED TO A DISCUSSION OF MEMORY MOVING AND FILE SAVING.

```
---------------------------------------
***************************************
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*      THE BASICS OF KRACKING 3:      *
*                             *
*   MEMORY MOVES, BINARY FILES, AND   *
*     KRAMMING FOR THE FINALS      *
*                             *
***************************************
```

IN THE LAST EPISODE, WE PONDERED THE STARTING ADDRESS OF A PROGRAM AND WAYS TO FIND IT IN SPITE OF THE PROTECTORS' SUBTERFUGE. THIS TIME WE'LL DISCUSS HOW TO GET THE PROGRAM INTO SAVEABLE FORMAT, EVEN IF IT'S TOO LONG TO SAVE AS A BFILE. ALTHOUGH WE'LL BE REFERRING AT FIRST TO SINGLE-LOAD PROGRAMS, MOST OF THESE TECHNIQUES ARE APPLICABLE TO PROGRAMS WITH DISK ACCESS.

BEFORE WE BEGIN THE PROCESS, LET ME PHILOSOPHIZE FOR A FEW SECONDS ON THE PROCEDURES AND PRACTICES TO BE USED. THIS IS A DISCIPLINE: PERHAPS NOT SO DEMANDING AS CHAMPIONSHIP KARATE OR THE UNIFICATION CHURCH, BUT IT REQUIRES KNOWLEDGE, PATIENCE, AND ATTENTION TO DETAIL. I URGE YOU TO BEGIN EACH ADVENTURE IN KRACKING WITH A SHARP PENCIL, PLENTY OF PAPER, AND A GOOD ERASER. FROM THIS POINT FORWARD IN OUR QUEST, RECORD-KEEPING WILL OCCUPY AN IMPORTANT PART OF THE TOTAL ACTIVITY. IF YOU HAVE A PRINTER, PRINT OUT ANY PERTINENT SECTIONS OF CODE AND WRITE IN YOUR OWN COMMENTS ABOUT WHAT IT MEANS. WRITE DOWN EVERY ADDRESS OF INTEREST, AND KEEP ESPECIALLY CAREFUL NOTES OF THE NATURE AND SEQUENCE OF ALL MEMORY MOVES, STARTING POINTS, AND TRICKS USED BY THE PROTECTORS. DO THIS NOT JUST BECAUSE IT'S CHARACTER BUILDING, BUT BECAUSE UNLESS YOU HAVE EXCEPTIONAL RECALL, ALL PROGRAMS WILL EVENTUALLY BLEND TOGETHER INTO A WARM AND FUZZY MEMORY. KEEP GOOD NOTES ON EVERYTHING YOU LEARN, AND REMEMBER: "THOSE WHO CANNOT RECALL THE MISTAKES OF THE PAST ARE DOOMED TO REPEAT THEM."

SUPPOSE YOU HAVE LOADED IN, RESET WITH YOUR OLD MONITOR ROM, AND FINALLY LOCATED THE STARTING ADDRESS TO THE GREATEST GAME EVER WRITTEN: "HYPERSPACE ANDROID CLONE KILLER" OR "HACK". THE STARTING ADDRESS IS 4123, AND THE GAME OCCUPIES MEMORY FROM 800 TO B000. YOU ALREADY KNOW THAT IF ANY MEMORY ABOVE 9D00 HAS BEEN USED BY THE PROGRAM, DOS IS DEAD, AND YOU CAN'T SAVE THE PROGRAM TO DISK WITH A DOS COMMAND. AS YOU ALSO UNDOUBTEDLY KNOW, IF THE PROGRAM WERE SMALLER YOU WOULD HAVE THE OPTION OF BOOTING A DISK AND SAVING THE GAME AS A BINARY FILE. LET'S TAKE JUST A SECOND, THOUGH, AND REVIEW WHAT HAPPENS TO MEMORY WHEN YOU BOOT A DISK.

FIRST OF ALL, DON'T USE A MASTER DISK, SINCE THE DOS ON A MASTER IS LOADED FIRST INTO 1600-3FFF AND THEN RELOCATED TO THE HIGHER REGIONS OF MEMORY. BOOTING A 48K SLAVE DISK WILL DISTURB ONLY 0-8FF AND 9600-BFFF, AND IF YOUR PROGRAM LIVES WITHIN OR CAN BE REARRANGED TO FIT THESE BOUNDARIES, YOU CAN

SAFELY BOOT THE DISK AND SAVE THE PROGRAM AS A BINARY FILE.

AN OLD METHOD OF SAVING A BINARY FILE IS WELL-KNOWN TO THOSE OF US WHO BOUGHT APPLES IN THE DARK AGES BEFORE THE DISK II, BUT THERE ARE NOW MAYBE HALF A MILLION (!) APPLE OWNERS WHO ARE UNFAMILIAR WITH THE CASSETTE PORT AND ITS USE. IN GENERAL, ALMOST ANY CASSETTE RECORDER THAT HAS A TONE CONTROL CAN BE USED, BUT FOR SOME REASON THE CHEAPER ONES ARE GENERALLY BETTER. TO USE ONE, PLUG BOTH CABLES INTO THE CORRECT CONNECTOR ("IN" MEANS INTO THE COMPUTER, NOT INTO YOUR RECORDER), AND TURN THE TONE CONTROL ALMOST TO THE TOP OF THE TREBLE RANGE. SAVE A SMALL BASIC PROGRAM (REFER TO THE MANUAL FOR USE OF THE BASIC COMMANDS) AT ANY OLD VOLUME CONTROL SETTING. TRY LOADING THE PROGRAM BACK IN SEVERAL TIMES, INCREASING THE VOLUME CONTROL SETTING UNTIL THE PROGRAM LOADS RELIABLY. YOU'LL FIND THAT THE TAPE WORKS VERY WELL, EVEN ON LONG FILES, ESPECIALLY WHEN THE SAME RECORDER IS USED TO RECORD AND PLAYBACK.

WHAT'S GOOD ABOUT THE TAPE SYSTEM IS THAT EVEN WHEN DOS IS COMPLETELY DEAD, THE MONITOR COMMANDS FOR TAPE I/O ARE STILL ACTIVE (ASSUMING YOU DIDN'T WIPE THEM OUT OF YOUR OLD MONITOR ROM). SEE THE REFERENCE MANUAL, PAGE 46 FOR A COMPLETE DESCRIPTION. WITH TAPE, YOU CAN ALWAYS SAVE ANY PART OF MEMORY AT ANY TIME! (WORTH KEEPING IN MIND FOR THOSE CRUCIAL SITUATIONS WHEN THE SYSTEM CRASHES JUST AS YOU ARE FINISHING YOUR TERM PAPER ON THE WORD PROCESSOR). THE CASSETTE ROUTINES USE ONLY LOCATIONS 3C-3F AND 42-43 IN ZERO PAGE, AND THE ONLY PART OF MEMORY YOU SHOULDN'T TRY TO SAVE IS C000-C0FF-- SOME TERRIBLE THINGS CAN HAPPEN IF YOU TRY. IN MOST CASES, IT'S BEST TO SAVE A LONG PROGRAM IN TWO FILES SO IT CAN BE RELOADED IN BETWEEN 800 AND 9600 AFTER DOS IS IN MEMORY. FOR OUR EXAMPLE OF "HACK", THE NECESSARY MONITOR COMMANDS ARE:

```
    *0.4FFFW    (LONG WAIT)
    *5000.AFFFW  (LONGER WAIT)
```

AFTER BOOTING A DISK, YOU CAN RELOAD WITH:

```
    *1000.5FFFR (RELOAD FIRST HALF)
    *BSAVE HACKLOW,A$1000,L$5000
    *BSAVE HACKHI,A$1000,L$6000
```

NOTE THAT IN THE TAPE READ AND WRITE COMMANDS, UNLIKE DOS, THE ACTUAL STARTING AND ENDING LOCATIONS ARE LISTED. BE SURE YOU UNDERSTAND THE ONE-BYTE DIFFERENCE BETWEEN THE TWO BEFORE YOU USE THEM.

THERE ARE ALSO OCCASIONS WHEN YOU WOULD LIKE TO SAVE APPLESOFT OR INTEGER BASIC PROGRAMS LOADED IN FROM A MODIFIED DOS ON A PROTECTED DISK (ARCADE MACHINE AND THE RAPID-FIRE SERIES FROM SSI ARE EXAMPLES). THIS IS SIMPLE WITH THE TAPE RECORDER, SINCE THE MONITOR ROUTINES ARE TOTALLY IGNORANT OF THE OPERATING SYSTEM IN RAM.   IF YOU CAN LIST A BASIC PROGRAM, YOU CAN USUALLY SAVE IT TO TAPE. TRY THE FOLLOWING WITH ONE OF THE ABOVE PROGRAMS: LOAD IN A PROGRAM MODULE (ANYTHING IN ARCADE MACHINE EXCEPT THE MAIN MENU), THEN HIT RESET WHILE IT'S RUNNING. TYPE D6:00 (THIS REMOVES THE APPLESOFT INTERNAL "PROTECTION"), THEN C081 TO SELECT THE MOTHER BOARD ROM (UNLESS YOU HAVE AN APPLE II WITH APPLESOFT ON A ROM CARD, THEN IT'S C080 TO SELECT SLOT 0). TYPE CONTROL-C AND YOU SHOULD BE ABLE TO LIST THE PROGRAM AND THEN SAVE IT TO TAPE WITH THE "SAVE" COMMAND (SOMETIMES AN ADDITIONAL FAIRLY TRIVIAL PROTECTION SCHEME IS USED WITH APPLESOFT PROGRAMS: DELETING THE FIRST LINE NUMBER SO IT WON'T LIST. IT WILL STILL SAVE TO TAPE AND YOU CAN RECONSTRUCT THE LINE NUMBER AT YOUR LEISURE). REMEMBER THAT THE BASIC "LOAD" AND "SAVE" COMMANDS DON'T ALLOW A FILE NAME TO BE ADDED.   IF THERE ARE MORE THAN A FEW FILES ON THE DISK, THIS IS A VERY TEDIOUS WAY TO KRACK A PROGRAM, BUT BACK IN THE MIDDLE AGES BEFORE DEMUFFIN PLUS IT WAS SOMETIMES THE ONLY WAY. YOU ALSO HAVE TO BE WARY OF BINARY ROUTINES WHICH ARE

CALLED FROM OR MODIFY THE BASIC PROGRAMS.

  YES, YOU'RE RIGHT. GETTING OUT AND HOOKING UP THE TAPE RECORDER IS A CRAMP IN THE CALVINS, SO IT'S USUALLY LEFT WORKS. IN GENERAL, IT'S BEST TO LEARN HOW TO MANIPULATE MEMORY TO SCRUNCH YOUR PROGRAM DOWN INTO A DOS FILE (IT WILL ALWAYS HAVE TO BE DONE, ANYWAY). IN THE BEST OF ALL POSSIBLE WORLDS, YOUR DOS WOULD BE IN ROM MEMORY, AND WOULD ALLOW YOU TO SAVE ANY PROGRAM THAT RESIDED IN RAM MEMORY. IN THE REAL WORLD, IT'S GENERALLY NECESSARY TO LOADED IN BY DOS FROM A NORMAL DISK (WE'LL TALK LATER ABOUT THOSE THAT CAN'T BE). THIS PROCESS IS USUALLY CALLED "MEMORY MOVING", AND THE PURPOSE IS TO "TUCK IN" ALL THE PIECES OF THE PROGRAM THAT LIE OUTSIDE THE NORMAL PROGRAM MEMORY OF 800-9600 ALLOWED BY DOS. THE OTHER HALF OF THE PROCESS IS THE "UNFOLDING" OF THE TUCKED-IN PORTIONS OF MEMORY AFTER THE PROGRAM IS RELOADED UNDER DOS. TO GAIN PERSPECTIVE ON THE PROCESS, LET'S LOOK AT MEMORY MAPS WITH DOS ACTIVE AND WITH "HACK" IN MEMORY.

```
_______________________________________
!                             !
!                             !
!F800-FFFF->!MONITOR ROM!AUTOSTART ROM!
!-------------------------------------!
!F000-F7FF->! INTEGER   ! APPLESOFT   !
!E800-EFFF->!       BASIC !     "       !
!E000-E7FF->!  "    "   !     "       !
!D800-DFFF->!(INSPECTOR)!     "       !
!D000-D7FF->! (WATSON)  !     "       !
!-------------------------------------!
!C800-CFFF->!PERIPHERAL SLOT ROM SPACE!
!C000-C7FF->!SOFT SWITCHES & SLOT ROMS!
!-------------------------------------!
!B800-BFFF->!           ^           !
!B000-B7FF->!           !           !
!A800-AFFF->!            DOS        !
!A000-A7FF->!           !           !
!9800-9FFF->!           V           !
!-------------------------------------!
!9000-97FF->!        ^              !
!8800-8FFF->!        !              !
!8000-87FF->!        !              !
!7800-7FFF->!        !              !
!7000-77FF->!        !              !
!6800-6FFF->! PROGRAM MEMORY        !
!6000-67FF->!        !              !
!------------------!------------------!
!5800-5FFF->!        !      ^       !
!5000-57FF->!        !  (HI-RES PAGE 2) !
!4800-4FFF->!        !      !       !
!4000-47FF->!        !      V       !
!------------------!------------------!
!3800-3FFF->!        !      ^       !
!3000-37FF->!        !  (HI-RES PAGE 1) !
!2800-2FFF->!        !      !       !
!2000-27FF->!        !      V       !
!------------------!------------------!
!1800-1FFF->!        !              !
!1000-17FF->!        !              !
!0800-0FFF->!        V  (TEXT PAGE 2)   !
!--------------------------------------
!0000-07FF->!ZERO PG,STACK,TEXT PAGE 1!
---------------------------------------
```

```
       AND, WITH "HACK" IN MEMORY:
_______________________________________
!         !       !           !
!F800-FFFF->!MONITOR ROM!AUTOSTART ROM!
!-----------------------------------!
!F000-F7FF->! INTEGER   ! APPLESOFT   !
!E800-EFFF->!       BASIC !     "       !
!E000-E7FF->!  "    "   !     "       !
!D800-DFFF->!(INSPECTOR)!     "       !
!D000-D7FF->! (WATSON)  !     "       !
!-----------------------------------!
!C800-CFFF->!PERIPHERAL SLOT ROM SPACE!
!C000-C7FF->!SOFT SWITCHES & SLOT ROMS!
!-----------------------------------!
!B800-BFFF->!           (EMPTY)           !
!B000-B7FF->!        ^              !
!A800-AFFF->!        !              !
!A000-A7FF->!        !              !
!9800-9FFF->!        !              !
!-----------------!-----------------!
!9000-97FF->!        !              !
!8800-8FFF->!        !  (EMPTY)           !
!8000-87FF->!        !  (EMPTY)           !
!7800-7FFF->!        !              !
!7000-77FF->!        !              !
!6800-6FFF->! PROGRAM "HACK"          !
!6000-67FF->!        !              !
!-----------------!-----------------!
!5800-5FFF->!        !      ^       !
!5000-57FF->!        !  (HI-RES PAGE 2) !
!4800-4FFF->!        !      !       !
!4000-47FF->!        !      V       !
!-----------------!-----------------!
!3800-3FFF->!        !      ^       !
!3000-37FF->!        !  (HI-RES PAGE 1) !
!2800-2FFF->!        !      !       !
!2000-27FF->!        !      V       !
!-----------------!-----------------!
!1800-1FFF->!        !  (EMPTY)           !
!1000-17FF->!        !  (EMPTY)           !
!0800-0FFF->!        V  (TEXT PAGE 2)   !
!-----------------------------------!
!0000-07FF->!ZERO PG,STACK,TEXT PAGE 1!
---------------------------------------
```

BEFORE WE BEGIN THE DISCUSSION OF THE TECHNIQUES OF MEMORY MOVING, LET'S RESTATE THE OBJECTIVE: WE'RE TRYING TO ARRANGE ALL THE PROGRAM INTO A SMALL ENOUGH SPACE THAT WE CAN BSAVE A FILE UNDER DOS (THE DOS MANUAL WILL TELL YOU THAT THE LARGEST BINARY FILE YOU CAN SAVE IS 128 SECTORS, BUT IF YOU CHANGE LOCATION $A964 (43364) TO $BF(191) YOU CAN SAVE A FILE AS LARGE AS THE ENTIRE RAM MEMORY). REMEMBER THAT BOOTING A SLAVE DISK WILL MESS UP 0-8FF AND 9600-BFFF, SO THE LARGEST FILE IT'S PRACTIVAL TO SAVE IS ABOUT 145 SECTORS (YOU CAN, WITH CARE, OVERWRITE MUCH OF THE SCREEN MEMORY AND PAGES 2 & 3 TO SAVE A BFILE OF ABOUT 151 SECTORS, BUT THAT REQUIRES KNOWLEDGE AND CONSIDERABLE CARE).

LOOKING AT THE MEMORY MAP WITH HACK, YOU CAN SEE THAT THE MEMORY FROM 9600 TO B000 WILL HAVE TO BE STORED SOMEWHERE ELSE TO BRING THE FILE SIZE DOWN, AND THE PAGE FROM 800-8FF WILL HAVE TO BE STASHED TEMPORARILY DURING THE DISK BOOT TO

RESTORE DOS. TO FIND OUT WHAT AREAS OF MEMORY ARE FREE, SEARCH THROUGH ALL MEMORY WITH THE INSPECTOR AND LOOK FOR BLANK PAGES. THE FOLLOWING TRICK WILL HELP: BEFORE YOU LOAD THE ORIGINAL, CLEAR ALL OF MEMORY TO ZERO (OR ANY OTHER BYTE YOU LIKE) WITH:

```
         *800:0
         *801<800.95FFM
```

THEN YOU'LL BE ABLE TO SEE UNUSED MEMORY AREAS. THIS DOESN'T ALWAYS WORK, SINCE MANY AREAS ARE COPIED TO A SECOND LOCATION AND NOT USED AFTERWARDS, SO IF YOU'RE HARD PRESSED FOR STORAGE MEMORY, IT'S A GOOD IDEA TO SCAN THROUGH ONCE WITH THE INPECTOR SET TO DECODE ASCII TO DETECT SUSPICIOUS SECTORS (LATELY, SOME OF THE PROTECTORS HAVE TAKEN TO STORING GARBAGE SUCH AS SOURCE CODE IN UNUSED PAGES OF MEMORY AND ON EMPTY DISK SECTORS). NOTE DOWN ANY PAGES THAT ARE TOTALLY CLEAR, ANY THAT ARE ALL ONE BYTE, REGARDLESS OF WHAT IS IS, OR ANY THAT CONTAIN JUNK. LET'S ASSUME FOR THIS EXAMPLE THAT LOCATIONS 1000-1FFF AND 8000-8FFF ARE BLANK. WE HAVE 1A00 (B000-9600) BYTES OF MEMORY "LEFTOVER" OR OUTSIDE OF THE DOS BOUNDARIES, SO THEY WILL ALL FIT INTO THE $2000 BLANK LOCATIONS THAT WE LOCATED.

STORE THE EXCESS BYTES IN THE HOLES BY TYPING:

```
     *8000<9600.A5FFM
     *1000<A600.AFFFM
```

OR EQUIVALENT; THE SPLIT CAN BE ANY WAY THAT HELPS YOU KEEP TRACK OF THE PROCESS. FINALLY, STASH THE MEMORY FROM PAGE 8 WITH *1B00<800.8FFM. REMEMBER THAT THIS IS ONLY TEMPORARY. BEFORE YOU DO ANYTHING ELSE, BOOT YOUR 48K SLAVE DISK, THEN RESTORE PAGE 8 WITH *800<1B00.1BFFM. BEFORE YOU DO ANYTHING ELSE, SAVE THE PROGRAM WITH "BSAVE HACKALL,A$800,L$8E00 (NINE OUT OF TEN TIMES YOU'LL FORGET TO CHANGE $A964; CONSIDER CHANGING IT IN THE DOS IN MEMORY BEFORE YOU INITIALIZE THE DISK SO IT WILL BE PERMANENT). YOU CAN NOW TAKE A DEEP BREATH AND RELAX: ALL OF THE PROGRAM MEMORY IS SAFELY TUCKED AWAY. ALL THAT'S LEFT IS TO WRITE A SHORT PROGRAM TO REVERSE THE MEMORY STORAGE.

TWO SHORT ROUTINES, SIMILAR TO THOSE SHOWN IN OUR FIRST BASICS LESSON ARE REQUIRED. AGAIN, LET'S REVIEW THE STEPS NECESSARY FROM HERE TO RUN THE GAME:

1. LOAD THE (COMPRESSED) GAME INTO 800-95FF.
2. MOVE THE PIECE OF MEMORY AT 8000-9FFF TO 9600-A5FF.
3. MOVE THE PIECE OF MEMORY AT 1000-19FF TO A600-AFFF.
4. JUMP TO THE STARTING ADDRESS AT $4123.

THE FOLLOWING PROGRAM WILL TAKE CARE OF STEPS 2-4. IT MAY NOT BE IMMEDIATELY OBVIOUS THAT THIS PROGRAM MUST BE STORED WITHIN THE COMPRESSED PROGRAM IN A PAGE THAT IS BOTH EMPTY AND UNAFFECTED BY THE MEMORY MOVES YOU ARE ABOUT TO MAKE. IN THIS CASE, PAGE 1C IS SAFE.

```
 1C00  LDY #$0       ;CLR Y-REG
 1C02  LDA $8000,Y ;GET A BYTE AT 8000+
 1C05  STA $9600,Y ;STORE IT AT 9600+
 1C08  INY     ;INCR. COUNTER
 1C09  BNE $1C02   ;IF NOT PAGEND, REDO
 1C0B  INC $1C04   ;INCR. SOURCE HIBYTE
 1C0E  INC $1C07   ;INCR. DEST HIBYTE
 1C11  LDA $1C07   ;GET THE DEST HIBYTE
 1C14  CMP #$90    ;IF 90,WE'RE DONE
 1C16  BNE $1C02   ;IF NOT, DO MORE
 1C18  LDA $1000,Y ;REPEAT THE PROCESS
 1C1B  STA $A600,Y ;FOR THE SECOND
 1C1E  INY     ;BLOCK
 1C1F  BNE $1CA8
 1C21  INC $1C1A
 1C24  INC $1C1D
 1C27  LDA $1C1D
 1C2A  CMP #$1B
 1C2D  BNE $1CA8
 1C2F  JMP 4123    ;AND JUMP TO THE
                 STARTING LOCATION
```

  THIS MAY SEEM HARD AT FIRST, BUT THE FORM IS SO CONSTANT THAT YOU'LL BE ABLE TO WRITE THESE MOVES IN YOUR SLEEP AFTER A FEW TRIES WITH THE MINI- ASSEMBLER (THE PLACE YOU'LL MOST LIKELY MESS UP IS IN THE 'CMP #90' BY TYPING 'CMP $90'--WATCH IT CAREFULLY!).

  TIME OUT FOR A BRIEF DISCUSSION OF ONE OF THE SUBTLE POINTS OF MEMORY MOVES. ALTHOUGH YOU'RE GENERALLY ABLE TO MAKE YOUR MEMORY MOVES NON-OVERLAPPING, YOU CAN HAVE A PROBLEM MOVING LARGE AMOUNTS OF MEMORY. THE MEMORY MOVE ROUTINES SHOWN ABOVE ARE "FORWARD" MEMORY MOVES: THAT MEANS THAT UST MOVED. SOMETIMES YOU WILL NEED TO MOVE, FOR INSTANCE, LOCATIONS 6000-8FFF TO 8000-AFFF. IF YOU USE THE FORWARD MOVES AS SHOWN, YOU CAN SEE THAT THE FIRST PAGE (PAGE 60 OR 6000-60FF) WILL LAND AT 8000-80FF, SMACK ON TOP OF THE ORIGINAL PAGE THAT WAS SUPPOSED TO BE MOVED LATER TO PAGE A0 (A000-A0FF). TO WORKS "DOWN" IN MEMORY INSTEAD OF UP.

  IN THIS EXAMPLE, PAGE 8F IS FIRST MOVED TO AF, THEN 8E TO AE, ETC. THIS WAY, WHEN IT FINALLY COMES TIMES TIME FOR PAGE 60 TO BE MOVED TO PAGE 80, THE ORIGINAL PAGE 80 WILL ALREADY HAVE BEEN MOVED. A TYPICAL ROUTINE FOR THIS IS:

```
 1000  LDY #$0
 1002  LDA $8F00,Y
 1005  STA $6000,Y
 1008  INY
 1009  BNE $1002
 100B  DEC $1004
 100E  DEC $1007
 1011  LDA $1007
 1014  CMP #$5F
 1017  BNE $1002
```

  OK--ALL THAT REMAINS IS TO GET TO THE START OF THE EARLIER MEMORY MOVE ROUTINE WHEN WE "BRUN" THE GAME. THIS IS ACCOMPLISHED BY PUTTING THE CODE FOR "JMP $1C00" OR 4C 00 1C AT LOCATION $7FD-$7FF AND MAKING THIS THE FIRST LOCATION OF THE PROGRAM. WE CAN THEN SAVE A COMPLETE, FUNTIONING VERSION OF HACK WITH "BSAVE HACK,A$7FD,L$8E03". THIS CREATES YOUR FINAL, 145-SECTOR FILE OF HACK WHICH WILL BRUN WHENEVER YOU WISH.

----------A FEW HELPFUL HINTS----------

  1. ALWAYS KEEP A FEW INITIALLIZED 48K SLAVE DISKS NEARBY--IT'S ALARMING HOW FAST A DISK FILLS UP WITH SLIGHTLY DIFFERENT 145-SECTOR VERSIONS OF THE PROGRAM UNDERGOING KRACKING.

  2. MAKE YOUR PROGRAM NAMES AS DESCRIPTIVE AS YOU CAN, ESPECIALLY WHEN SAVING A PROGRAM IN PIECES. IT'S VERY DISTURBING TO RETURN TO A KRACKING EFFORT AFTER A LONG WEEKEND TO FIND PROGRAMS ON THE DISK TITLES "HACKHI", "HACKHIGH", "HIGH",

"HH", ETC. AND NOT BE SURE WHAT EACH ONE IS. BETTER TO TYPE IN A FEW EXTRA LETTERS TO LET YOU KNOW THAT IT'S "HACK WITHOUT 9600UP" OR OR "HACK 4000-B000 ONLY".

3. WHENEVER POSSIBLE, COMPRESS THE GAME TO THE MINIMUM NUMBER OF SECTORS BY DOING A FEW MORE MEMORY MOVES BEFORE AND AFTER SAVING. YOUR FRIENDS WILL APPRECIATE YOUR THOUGHTFULNESS IN MAXIMIZING THE NUMBER OF GAMES PER DISK AND MINIMIZING MODEM TIME.

4. =>VERY IMPORTANT<= WHEN YOU THINK YOU HAVE A COMPLETE, WORKING VERSION, CHECK IT OUT THOROUGHLY ON ALL LEVELS AND IN ALL MODES. IT'S EXTREMELY EMBARASSING TO HAVE TO ISSUE A "PRODUCT RECALL" WHEN YOU LEARN A MONTH LATER THAT HACK CRASHES ON LEVEL 47 JUST AS THE HYPERGALACTIC FROG IS ABOUT TO DEVOUR NEW PITTSBURGH ON THE MARS COLONY...

```
*** NEXT TIME--PICTURE PACKING AND ***
********* RAM CARD TECHNIQUES ********

---------------------------------------
***************************************
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*      BASICS OF KRACKING - 104       *
*                             *
***************************************
```

WHERE DO I BEGIN?

SEVERAL PREVIOUS EPISODES OF THIS COLUMN HAVE DEALT WITH THE RELATIVELY SIMPLE TECHNIQUES WHICH CAN BE USED TO SAVE A SINGLE-LOAD FILE TO DISK AS AN UNPROTECTED BINARY PROGRAM, AND IT IS NOW TIME TO EXPLORE THE LARGER AREA OF MULTIPLE-PROGRAM DISKS, PROGRAMS WITH DISK ACCESS, AND THE APPROACHES USED TO PROTECT THEM FROM BEING COPIED. WE WILL BEGIN WITH SIZING UP A DISK PROTECTION SCHEME, DECIDING ON A BASIC APPROACH, AND BEGINNING THE UNPROTECTION PROCESS (THE SUBJECT OF BOOT-TRACING AS ANOTHER MEANS TO THE SAME END WILL BE DESCRIBED IN A FUTURE EPISODE, SINCE IT IS GENERALLY USED WITH MORE SOPHISTICATED PROTECTION SCHEMES). THE SUBJECT IS TRULY MAMMOTH, AND WILL REQUIRE SEVERAL EPISODES TO COMPLETE. FOR NOW, SETTLE BACK, OPEN A COLD BEVERAGE OF YOUR CHOICE, AND LET'S BEGIN A JOURNEY INTO THE FIRST LEVEL OF DISK PROTECTION: THE MODIFIED DOS (AS WE HAVE OFTEN MENTIONED BEFORE, TWO STALWART FRIENDS IN THIS QUEST ARE "BENEATH APPLE DOS" BY WORTH AND LECHNER, AND RANDY HYDE'S "DOSSOURCE". IT IS POSSIBLE TO KRACK DISKS WITHOUT THEM, BUT WITH ABOUT THE SAME EASE AS PERFORMING AN ORAL APPENDECTOMY).

APPLE'S DOS, COMBINED WITH THE DIVISION OF HARDWARE BETWEEN THE DISK CONTROLLER CARD AND THE DISK ANALOG BOARD, IS A VERITABLE PLAYGROUND FOR THOSE WHO PRODUCE DISK PROTECTION. THERE ARE LITERALLY THOUSANDS OF DIFFERENT THINGS WHICH CAN BE DONE TO MAKE COPYING A DISK DIFFICULT, CHALLENGING, AND (MAYBE SOMEDAY), IMPOSSIBLE. IN SO DOING, THEY PROVIDE HOURS OF VERY INGENIOUS PUZZLES, BOUNDLESS INTELLECTUAL STIMULATION, AND NOT INCIDENTALLY, THE INCENTIVE TO LEARN MUCH MORE ABOUT PROGRAMMING, THE APPLE, DOS, ASSEMBLY LANGUAGE, AND TREACHERY THAN WE WOULD OTHERWISE HAVE THE DESIRE TO LEARN.

BY FAR THE MOST COMMON TECHNIQUE USED TO PROTECT ENTIRE DISKS IS TO MAKE MODIFICATIONS TO THE OPERATING SYSTEM, AND SPECIFICALLY TO THE READ/WRITE TRACK AND SECTOR (RWTS) ROUTINES WHICH DEFEAT ORDINARY COPY PROGRAMS (COPYA AND SUPER DISK COPY 3.X ARE EXAMPLES, BUT WE'LL SEE LATER HOW BOTH OF THESE CAN BE USED TO

OUR ADVANTAGE). TO FIND THE MOST EFFICIENT APPROACH TO DEFEATING THESE PROTECTION TECHNIQUES, WE NEED FIRST TO SPEND A FAIR AMOUNT OF TIME DESCRIBING IT FROM THE CRACKIST'S VIEWPOINT.

(ON THE FUNDAMENTAL PRINCIPLE THAT GIVING A MAN A FISH ALLOWS HIM TO EAT FOR A DAY WHILE TEACHING HIM TO FISH ALLOWS HIM TO EAT FOR LIFE, WE WILL NOT DWELL ON THE SUBJECT OF "COPYING" AS SUCH. MANY OF THE TECHNIQUES DESCRIBED HERE ARE, HOWEVER, VERY USEFUL IN DECIDING HOW TO GO ABOUT COPYING A DISK. PERHAPS AN ASPIRING AUTHOR OUT THERE WILL BUILD FROM THE INTRODUCTION GIVEN HERE TO PURSUE THE SUBJECT IN DEPTH...?)

BEFORE WE CAN GET TO THE CORE OF THE MATTER, WE MUST UNDERSTAND MUCH MORE OF THE PROCESSING AND ENCODING SYSTEMS USED BY DOS TO STORE INFORMATION ON THE DISK. THIS IS FAIRLY HEAVY STUFF, BUT YOUR KRACKING ABILITY DEPENDS MORE THAN ANYTHING ELSE ON YOUR KNOWLEDGE OF THIS SUBJECT. TRY YOUR BEST TO WORK THROUGH IT NOW, AND THE REST OF THE PROCESS WILL BE MUCH EASIER.

WE ALREADY KNOW THAT EACH TRACK CONSISTS OF 16 SECTORS WHICH EACH REPRESENT ONE PAGE (256 BYTES) OF DATA. A SECTOR ACTUALLY CONSISTS OF TWO SEPARATE PARTS, AN ADDRESS FIELD, WHICH TELLS DOS WHICH SECTOR IT IS, AND A DATA FIELD, WHERE THE ACTUAL BYTES ARE STORED. TO BEGIN A TRIP AROUND THE DISK, LET'S LOOK FIRST AT THE BYTE SEQUENCE TAKEN FROM A NORMAL, UNMODIFIED DOS DISK AT TRACK0, SECTOR 0 (AS WE MENTIONED EARLIER, THE TERMS 'BYTE' AND 'NIBBLE' ARE OFTEN USED INTERCHANGEABLY TO REFER TO THE DATA READ OFF THE DISK. THE USE OF 'NIBBLE' IS NOT REALLY ACCURATE IN REFERENCE TO DOS 3.3, BUT PERSISTS FOR HISTORICAL REASONS).

```
----FF FF FF FF D5 AA 96 FF FE AA AA ->
            / \      / \   / \   /
________(1)__/     \_(2)/   (3)     (4)

->AA AA FF FE DE AA EB FF FF FF FF ----
  \   / \   / \   /     \
   (5)       (6)   (7)  \_____(8)______
```

THE FIRST FEW FF'S (1) ARE KNOWN AS GAPBYTES, BUT THEY'RE CORRECTLY TERMED SYNCBYTES, AND WE'LL TREAT THEM AS SIMPLE SEPARATORS FOR NOW. NEXT ARE THE THREE MOST IMPORTANT BYTES ON THE DISK, D5 AA 96 (2). THIS SEQUENCE MAY NOT OCCUR ANYWHERE ELSE ON THE DISK EXCEPT THE ADDRESS FIELD, AND SERVES AS A UNIQUE IDENTIFICATION MARKER. THESE BYTES ARE KNOWN BY ALL SORTS OF COLLOQUIALISMS, INCLUDING "ADDRESS MARKER", "HEADER BYTES", "LEADER BYTES", "PROLOG", AND OTHERS. THEY WILL ALWAYS, REPEAT ALWAYS, OCCUR ON AT LEAST SECTOR 0 OF TRACK 0 OF EVERY APPLE DISK WHICH LOADS UNDER DOS 3.3 (THE FIRST LAW DEMANDS IT).

THE NEXT FOUR SEQUENCES ENCODE THE VOLUME NUMBER (3), TRACK NUMBER (4), SECTOR NUMBER (5), AND CHECKSUM (6). EACH NUMBER IS A SINGLE BYTE, WRITTEN IN AN OLD-STYLE ENCODING SCHEME CALLED 4+4 NIBBLIZING. THIS IS A FORMAT FOR STORING DATA ON THE DISK IN WHICH THE EVEN BITS OF A BYTE ARE STORED IN ONE 8-BIT SEQUENCE (REPRESENTING ONE-HALF OF THE ORIGINAL BYTE OR ONE NIBBLE), AND THE ODD BITS ARE STORED IN THE SECOND "BYTE" (THE REQUIREMENT FOR THIS SORT OF "BYTE-SPLITTING" OR NIBBLIZING WAS ESTABLISHED LARGELY BY THE LIMITATIONS IMPOSED BY DISK DRIVE HARDWARE. YOU CAN FIND MUCH MORE INFORMATION IN B. A. D., PP. 3-12 TO 3-21, BUT AN OVERSIMPLIFICATION IS THAT, IN THE OLD DAYS, AT LEAST EVERY OTHER BIT READ FROM THE DISK HAD TO BE A LOGICAL "ONE", OR THE CIRCUITRY THAT READ THE DISK "FORGOT" WHERE IS WAS AND WHAT IT WAS DOING). IF YOU ARE INTERESTED IN MORE DETAIL ON THE MECHANICS OF THE 4+4 SCHEME, REFER TO THE VERY FIRST KRACKING KORNER FILE ON CYCLOD AND THE FILE ON WAY OUT. THE TABLE BELOW LISTS THE VALUES OF NIBBLES OF INTEREST TO US IN THIS FORMAT:

FIRST NIBBLE	SECOND NIBBLE	BYTE VALUE	FIRST NIBBLE	SECOND NIBBLE	BYTE VAL.
AA	AA	0	AA	BA	10
AA	AB	1	AA	BB	11
AB	AA	2	AB	BA	12
AB	AB	3	AB	BB	13
AA	AE	4	AA	BE	14
AA	AF	5	AA	BF	15
AB	AE	6	AB	BE	16
AB	AF	7	AB	BF	17
AE	AA	8	AE	BA	18
AE	AB	9	AE	BB	19
AF	AA	A	AF	BA	1A
AF	AB	B	AF	BB	1B
AE	AE	C	AE	BE	1C
AE	AF	D	AE	BF	1D
AF	AE	E	AF	BE	1E
AF	AF	F	AF	BF	1F
			BA	AA	20
			BA	AB	21
			BB	AA	22
			FF	FE	FE
				(VOL#254)	

WE CAN NOW DECODE THE FOUR GROUPS OF BYTES AS: VOL# 254 (3), TRACK# 0 (4), SECTOR# 0 (5), AND CHECKSUM $FE (6). THE FIRST THREE ARE SELF- EXPLANATORY, AND THE LAST IS USED TO DETECT ANY ERRORS WHICH MAY CREEP IN AFTER MANY HOURS OF DISK USE. FOLLOWING THESE IS A SEQUENCE OF BYTES (7) USED TO MARK THE END OF THE ADDRESS FIELD. A TOTAL OF THREE BYTES (DE AA EB) ARE WRITTEN TO THE DISK, BUT ONLY THE FIRST TWO ARE CHECKED WHEN THE FIELD IS READ. THIS PAIR OF BYTES IS KNOWN VARIOUSLY AS "CLOSING BYTES", "TRAILERS", OR THE "EPILOG". FINALLY, THERE IS ANOTHER SERIES OF GAPBYTES (8) WHICH SEPARATES THE ADDRESS FIELD FROM THE FOLLOWING DATA FIELD.8

THE DATA FIELD HAS A SIMILAR STRUCTURE:

```
----FF FF FF D5 AA AD ----342 BYTES OF
         / \          / \
______(1)_/   \_(2)/   \_____(3)____->

DATA----(CKSUM) DE AA EB FF FF FF----
   /  \     / \   /         \
<-/    \(4)/       (5)       \__(6)__
```

WHERE THE GAPBYTES (1) ARE SAME GROUP THAT ENDED THE ADDRESS FIELD. THE DATA MARKER BYTES (2) ARE ALSO CALLED BY ALL THE NAMES MENTIONED FOR THE ADDRESS MARKER, AND ARE INTERPRETED BY DOS AS "HERE COMES THE DATA..." THE BIG STRETCH OF 342 BYTES (3) IS A VERY COMPLEX WAY OF STORING 256 BYTES ON A DISK, FOLLOWING SOME COMPROMISES MADE WITH THE ORIGINAL LAWS OF DISK RECORDING. WITHOUT GOING INTO EXACTLY WHY, EACH "BYTE" CAN REPRESENT ONLY 6 BITS OF AN ORIGINAL BYTE, WHICH MEANS THAT EACH BYTE HAS TWO BITS LEFT OVER. PACKING THESE TOGETHER AT 6 BITS EACH REQUIRES ANOTHER 256/3 OR 86 DISKBYTES, FOR A TOTAL OF 256+86=342 "BYTES", WHICH NO LONGER REPRESENT A NIBBLE OR HALF A BYTE, BUT 3/4 OF A BYTE

(MAKE UP YOUR OWN NAME FOR IT, THERE'S NO REAL AGREEMENT WHAT IT SHOULD BE CALLED).

FOLLOWING THE DATA IS A SINGLE CHECKSUM BYTE (4), WHICH WILL GIVE ZERO WHEN EXCLUSIVE-ORED WITH ALL THE OTHER BYTES FROM THE DATA, AND THEN THE SAME ACTIVE CLOSING BYTES THAT WERE USED IN THE ADDRESS FIELD (5).   FINALLY, MORE GAPBYTES (6) PAD THE SPACE BETWEEN THIS DATA FIELD AND THE ADDRESS FIELD WHICH COMES NEXT.

THIS SEQUENCE IS REPEATED 15 MORE TIMES TO MAKE A COMPLETE TRACK, AND THERE IS USUALLY A LARGE "GAP" OF UP TO 128 FF'S SEPARATING THE LAST AND THE FIRST SECTORS ON THE TRACK. ONE FINAL ITEM OF INTEREST IS THAT THE SECTORS DO NOT NORMALLY FOLLOW EACH OTHER IN NUMERICAL SEQUENCE OF 0,1,2...ETC. THE ACTUAL SEQUENCE (CALLED "SKEWING" OR INTERLEAVING) IS CHOSEN FOR SPEED OF READING AND WRITING, AND CAN VARY ON SOME DOS'S WHICH ARE OTHERWISE STRICTLY IDENTICAL IN FORMAT TO DOS 3.3.

THIS WOULD BE A GOOD POINT, IF YOU'RE NOT ALREADY VERY COMFORTABLE WITH THE SEQUENCES DESCRIBED ABOVE, TO GET OUT A UTILITY WHICH WILL PERFORM A "NIBBLE READ" OF A DISK TRACK (INSPECTOR, NIBBLES AWAY, LOCKSMITH, ETC.), AND READ IN A STANDARD DOS TRACK. SCAN THROUGH THE BYTES UNTIL YOU COME TO THE MAGICAL D5 AA 96 SEQUENCE, THEN COMPARE ALL THE BYTES WHICH FOLLOW IT TO THE DESCRIPTION GIVEN ABOVE.   TRY A FEW TRACKS AND DECODE THE START OF SEVERAL SECTORS UNTIL YOU BECOME FAMILIAR WITH THE APPEARANCE OF THEM. YOU'LL SAVE YOURSELF A LOT OF TIME AND EFFORT BY BECOMING FAMILIAR WITH THE APPEARANCE OF NORMAL DOS SECTORS AND TRACKS.

KNOWING THAT ALL THESE THINGS ARE REQUIRED TO MAKE A DISK COMPATIBLE WITH DOS 3.3 (AND MAKE IT COPY WITH COPYA), YOU CAN EASILY SEE HOW TO MAKE A PROTECTED OR MODIFIED DOS: SIMPLY CHANGE ALMOST ANY ONE OF THE IMPORTANT BYTES IN EITHER OR BOTH FIELDS, AND MAKE THE APPROPRIATE CHANGES TO THE READ AND WRITE ROUTINES IN DOS. IN ORDER TO APPRECIATE WHAT THIS MEANS, LET'S SPEND A MINUTE OR TWO ON THE STRUCTURE OF DOS.

JUST AS WAS GAUL, DOS IS DIVIDED INTO THREE MAIN PARTS. THE FIRST ONE, CALLED THE COMMAND INTERPRETER, HAS BEEN DESCRIBED IN CONSIDERABLE DETAIL BY BERT KERSTHE DISK UNDER ANYTHING RESEMBLING A NORMAL DOS. AFTER A COMMAND (KEYBOARD OR PROGRAM) HAS BEEN PROCESSED BY THE COMMAND INTERPRETER, AND THE RIGHT PART OF THE RIGHT FILE HAS BEEN SELECTED BY THE FILE MANAGER, THE RWTS ROUTINES ARE CALLED ON TO DO THE CRUCIAL JOB OF EXCHANGING INFORMATION BETWEEN THE APPLE'S MEMORY AND THE DISKETTE.

SPACE PREVENTS US FROM LISTING ALL THE ROUTINES, BUT THOSE OF PARTICULAR INTEREST ARE:

```
 ADDRESS  NAME          FUNCTION
 -------  ----      --------------------
B700-B749 -- DO 2ND STAGE BOOT LOAD, RUN HELLO PROG.

B793-B7B4 RWPAGES READ OR WRITE A GROUP OF PAGES

B7B5-B7C1 CALLRWTS DISABLE INTERRUPT AND CALL RWTS

B7E8-B7F8 IOBLOCK RWTS PARMLIST-SEE B.    A.  D.

B800-B829 PRENIB CONVERT BYTES TO NIB- BLES FOR WRITING

B82A-B8C1 WRITE WRITE SECTOR TO DISK
```

```
B8C2-B8DB POSTNIB CONVERT NIBBLES TO BYTES AFTER READING

B8DC-B943 READ READ SECTOR FROM DISK

B944-B99F RDADR READ AN ADDRESS FIELD

B9A0-BA28 SEEKABS POSITION READ HEAD TO THE DESIRED TRACK

BA29-BA68 NIBL WRITE TRANSLATE TABLE

BA69-BA95 (EMPTY) =>WATCH THIS SPACE<=

BA96-BAFF -- READ TRANSLATE TABLE

BB00-BBFF NBUF1 BUFFER (PRIM) USED TO STASH THE NIBBLES

BC00-BC55 NBUF2   SEC. BUFFER FOR NIBLS

BC56-BCC3 WRADR WRITE ADDRESS FIELD (ONLY DURING INIT)

BCDF-BCFF (EMPTY) =>BE SUSPICIOUS<=

BD00-BDEC RWTS      MAIN READ/WRITE ORGN

BDED-BE03 RDRIGHT GOOD READ, CK TRACK #

BE10-BE25 RTTRK   RIGHT TRK, CK VOL#

BE26-BE45 CRCTVOL RIGHT VOL#, CK SECT#

BEAF-BFB7 DSKFORM INITIALLIZE DISK

BFD8-BFC7 SECMAP  SECTOR INTERLEAVE MAP

BFC8-BFFF PATCHES CORRECTIONS FOR SMALL DOS BUGS =>BEWARE<=
=======================================
```

AS BEFORE, YOU ARE STRONGLY URGED TO GET AS FAMILIAR AS YOU CAN WITH THESE ROUTINES, USING DOSSOURCE AND B. A. D. AS YOUR PRIMARY REFERENCES.

RETURNING TO THE SUBJECT OF DETECTING AND CIRCUMVENTING MODIFIED DOS'S, YOU HAVE A CHOICE. YOU CAN EITHER LOOK FOR CHANGES BY INSPECTING A TRACK, OR YOU CAN SEARCH THROUGH THE ABOVE RWTS ROUTINES FOR SOMETHING THAT ISN'T NORMAL. NEITHER APPROACH WILL WORK 100% OF THE TIME, SO IT'S BEST TO BECOME PROFICIENT AT BOTH. THE TABLE BELOW LISTS MOST OF THE CRUCIAL LOCATIONS IN RWTS THAT ARE COMMONLY CHANGED FOR THE PURPOSE OF PROTECTION.

```
      NORMAL
ADDRESS VALUE      USE
------- ------ -----------------------
 B853   D5   DATA ADDR MARKER 1-WRITE
 B858   AA   DATA ADDR MARKER 2-WRITE
 B85D   AD   DATA ADDR MARKER 3-WRITE
 B89E   DE   EPILOG BYTE 1
 B8A3   AA   EPILOG BYTE 2
 B8A8   EB   EPILOG BYTE 3-NOT READ
 B8AC   FF   EPILOG BYTE 4-NOT READ
 B8E7   D5   DATA ADDR MARKER 1-READ
 B8F1   AA   "   "     "    2   "
 B8FC   AD   "   "     "    3   "
B92A-C D9 00 BA LOCATION FOR CHECKSUM
            COMPARE

 B935   DE   EPILOG BYTE 1-READ
 B93F   AA   EPILOG BYTE 2-READ

 B942   38   SET CARRY FOR I/O ERROR

 B955   D5   ADDR DATA MARKER 1-READ
 B95F   AA   "    "     "   2  "
 B96A   96   "    "     "   3  "

 B991   DE   ADDR EPILOG BYTE 1
 B99B   AA   ADDR EPILOG BYTE 2

BA29-68   *    WRITE TRANSLATE TABLE

BA96-FF   *    READ TRANSLATE TABLE

 BC5F   FF   SYNC BYTE VALUE

 BC7A   D5   ADDR MARKER WRITE-1
 BC7F   AA   ADDR MARKER WRITE-2
 BC84   96   ADDR MARKER WRITE-3

 BCAE   DE   ADDR EPILOG BYTE 1-WRITE
 BCB3   AA   "     "     "  2   "
 BCB8   EB   "     "     "  3   "

BFB8-C7   *    SECTOR INTERLEAVING
             TABLE
```

* SEE DOSSOURCE LISTING FOR CORRECT CONTENTS.

ANY OF THE LOCATIONS ABOVE CAN BE MODIFIED, EITHER PERMANENTLY (WHICH CHANGES IN THE DOS IMAGE ON TRACKS 0-2), OR TEMPORARILY. THE TEMPORARY DOS CHANGES ARE MUCH TOUGHER TO FIND THAN THE PERMANENT ONES, SINCE THE CHANGES MAY BE ERASED AFTER THEY HAVE BEEN USED. A GOOD EXAMPLE OF THIS WAS 'MASK OF THE SUN' AND 'THE SERPENT'S STAR', WHERE THE MAIN DISK IS PROTECTED (AMONG OTHER TECHNIQUES) BY USING FF'S FOR ALL THE EPILOG BYTES, BUT THE SAVE GAME IS WRITTEN OUT AND READ IN USING THE NORMAL DE AA'S. A PAIR OF SUBROUTINES WAS CALLED TO SWAP THE BYTES IN AND OUT AS REQUIRED. MUCH MORE DEVIOUS WAS THE PROTECTION SCHEME USED BY TSR ON 'COMPUTER DUNGEON' AND 'THESEUS AND THE MINOTAUR', WHERE EPILOG BYTES WERE COMPUTED ACCORDING TO WHICH TRACK WAS BEING READ.

THERE ARE MANY OTHER EXAMPLES OF DOS MODIFICATIONS USED TO KEEP US AT BAY, INCLUDING SOME SECONDARY PROTECTION TECHNIQUES, BUT WE'LL LOOK AT THOSE AFTER WE DESCRIBE THE BASIC APPROACHES TO UNPROTECTING THESE DISKS IN PART 5 OF THE BASICS OF KRACKING. IN THE MEANTIME, STUDY THE FORMAT, BECOME FAMILIAR WITH THE STANDARD TRICKS, AND REMEMBER:

"THE NIGHT SHALL BE FILLED WITH MUSIC,
     AND CARES THAT INFEST THE DAY,
 SHALL FOLD THEIR TENTS LIKE THE ARABS,
     AND AS SILENTLY, STEAL AWAY."

```
          -HENRY WADSWORTH LONGFELLOW

---------------------------------------
***************************************
*                             *
*                             *
*    KRAKOWICZ'S KRACKING KORNER      *
*                             *
*                             *
*      THE BASICS OF KRACKING 5       *
*                             *
* UNPROTECTION OF MODIFIED DOS DISKS  *
*                             *
*                             *
***************************************
```

IN EPISODE 4 OF THIS SERIES, WE BEGAN A DISCUSSION OF PROTECTION SCHEMES WHICH ARE BASED ON MODIFICATION OF A STANDARD APPLE DOS. AS WE MENTIONED, THERE ARE MANY CHANGES WHICH CAN BE MADE, AND LITERALLY THOUSANDS OF COMBINATIONS OF WHICH CAN BE USED TO THWART THE STANDARD COPY PROGRAMS. HOWEVER, RATHER THAN DWELLING ON ALL THE POSSIBLE TECHNIQUES, LET'S CONCENTRATE ON THE "SHOTGUN" APPROACH WHICH WORKS TO THE VAST MAJORITY.

REGARDLESS OF THE MODIFICATION TECHNIQUE USED, MOST OF THESE DISKS CAN BE RENDERED COPYABLE WITH SOME UTILITY PROGRAMS (BOTH OLD AND NEW).

IN GENERAL, IT IS POSSIBLE TO IDENTIFY DISKS WITH A MODIFIED DOS BY THE APPEARANCE OF A BASIC PROMPT AT THE BOTTOM OF THE SCREEN DURING THE BOOT. SOME PROTECTORS HAVE BEGUN TO BYPASS THE ROUTINE WHICH OUTPUTS THE PROMPT, BUT YOU CAN STILL GUESS THAT THERE'S A MODIFIED DOS PRESENT IF THE BOOT SOUNDS LIKE A NORMAL DOS BOOT, BUT THE DISK WON'T COPY WITH COPYA (COMPARING THE SOUNDS MADE BY THE BOOT UNDER DIFFERENT PROTECTION SCHEMES CAN BE VERY VALUABLE AFTER YOU HAVE A FAIR AMOUNT OF EXPERIENCE WITH A GIVEN PUBLISHER AND HIS PROTECTION SCHEME. IT CAN ALSO BE MISLEADING; I KNOW A LOT OF PEOPLE WHO SWORE THE LONG HEAD MOVE DURING THE BOOT OF THE SSI RDOS DISKS WAS A NIBBLE COUNT, WHILE IT TURNED OUT TO BE NOTHING MORE THAN LOADING IN A SHORT PROGRAM CALLED "QWERTY" FROM TRACK 18-22).

THE CLASSIC PROGRAM FOR DEALING WITH MODIFIED DOS'S IS CALLED DEMUFFIN PLUS (WILL THE REAL AUTHOR PLEASE STEP FORWARD SOMEDAY TO ACCEPT THE THANKS OF THE ENTIRE WORLD OF SOFTWARE UNPROTECTORS?), AND IT WORKS IN MUCH THE SAME WAY AS APPLE'S MUFFIN PROGRAM. MUFFIN WAS WRITTEN TO READ FILES FROM A DOS 3.2 DISK AND THEN WRITE THEM OUT IN 3.3 FORMAT.   DEMUFFIN (AND A SIMILAR PRODUCT CALLED "NIFFUM") WERE WRITTEN TO CONVERT DOS 3.3 PROGRAMS TO 3.2 FOR THE REAL DIEHARDS. DEMUFFIN PLUS OPERATES ON THE SAME PRINCIPLE, BUT USES WHATEVER DOS IS IN MEMORY TO READ, THEN WRITES OUT TO AN INITIALLIZED DISK UNDER 3.3 FORMAT. WHILE THIS IS A POWERFUL UTILITY, YOU MUST KEEP IN MIND THAT IT IS STRICTLY BASED ON DOS, AND WILL ONLY TRANSFER PROGRAMS WHICH CAN BE LOCATED FROM A CATALOG AS NORMAL TEXT, BINARY, INTEGER OR APPLESOFT FILES.

IT IS SAFE TO SAY THAT MORE SOFTWARE HAS BEEN UNPROTECTED WITH THIS UTILITY THAN WITH ANY OTHER, AND PROBABLY MORE THAN WITH ALL OTHERS COMBINED. IT STILL FINDS FREQUENT APPLICATION TODAY, SO WE'LL TAKE A LITTLE TIME HERE TO DESCRIBE SEVERAL WAYS OF USING IT.

IN MOST CASES, THE CLASSICAL TECHNIQUE WHICH FOLLOWS FOR USING DEMUFFIN PLUS TO KRACK A MODIFIED DOS DISK IS RESTRICTED TO THOSE WITH AN APPLE II (NOT A II+)

OR A ROMCARD OR OTHER MODIFED F8 ROM WHICH ALLOWS YOU TO RESET INTO THE MONITOR (SEE THE BASICS OF KRACKING 1):

1. INITIALLIZE A DISK UNDER DOS 3.3, THEN DELETE THE HELLO PROGRAM (JUST TO BE SAFE).

2. BLOAD DEMUFFIN PLUS, A$6000.

3. BOOT THE PROTECTED DISK, AND AS SOON AS A PROMPT APPEARS, HIT RESET (GENERALLY ABOUT 3-4 SECONDS AFTER THE HEAD CLACKETY-CLACK).

4. FROM THE MONITOR, MOVE DEMUFFIN PLUS TO ITS NORMAL LOCATION WITH 803<6000.78FFM.

5. TYPE 803G AND FOLLOW THE FAMILIAR FIDDISH INSTRUCTIONS FOR SLOT AND DRIVE USAGE.

6. IF YOU WANT THE DISK TO AUTORUN FROM THE BOOT, DETERMINE THE NAME OF THE HELLO PROGRAM AND ENTER IT INTO TRACK 1, SECTOR 9, BYTES 75-92 WITH THE INSPECTOR (OTHERWISE THE DISK WILL ALWAYS LOOK FOR AN ACTUAL PROGRAM NAMED 'HELLO'). IF THE HELLO PROGRAM HAPPENS TO BE A BINARY FILE, CHANGE BYTE 42 IN TRACK 0, SECTOR D TO $34, OR USE $14 TO EXEC A TEXT FILE FOR THE START.

IN ADDITION TO HIDING THE PROMPT, A NUMBER OF PUBLISHERS HAVE ADDED ROUTINES WHICH CLEAR OUT MEMORY DURING THE BOOT, OR LOOK FOR SPECIFIC DATA LOADED IN PREVIOUSLY (THE PLATO SERIES IS A GOOD EXAMPLE OF THIS--WITHOUT EXTENSIVE AND CAREFUL BOOT TRACING, IT IS VERY DIFFICULT TO GET THE DOS IN MEMORY INTACT). THE FOLLOWING APPROACH ELIMINATES THE NEED TO RESET INTO THE MONITOR, AND ALSO GETS AROUND MANY OF THE ROUTINES BEING ADDED TO THWART THOSE WHO WOULD RESET. BECAUSE THERE IS NO NEED TO RESET DURING THE BOOT, THIS APPROACH CAN BE USED ON ANY FLAVOR OF APPLE][(YES, NIBBLESPOCK, EVEN ON A IIE). BASICALLY, THE DIFFERENCE LIES IN USING THE COMMAND INTERPRETER AND FILE MANAGER PORTIONS OF A STANDARD DOS, AND ADDING TO IT THE MODIFIED RWTS FROM THE PROTECTED DISK:

1. INITIALLIZE A DISK AS BEFORE.

2. BOOT UP A STANDARD DOS DISK, AND BLOAD DEMUFFIN PLUS,A$4000 (JUST TO BE SAFE).

3. UNLESS YOU HAVE INSPECTOR IN ROM, BLOAD INSPECTOR,A$8800.

4. USING THE INSPECTOR, READ TRACK 0, SECTOR 1 THROUGH TRACK 0, SECTOR 9 INTO $7700 TO $7FFF. USE CONTROL-I AFTER THE FIRST 'R' COMMAND TO SPEED THE LOAD (SEE THE SECTION BELOW ON CHANGING LOCATION $B942 IF YOU CAN'T READ THE SECTORS WITH THE INSPECTOR).

5. GET INTO THE MONITOR, THEN MOVE THE RWTS THAT YOU JUST READ IN ON TOP OF THE RESIDENT RWTS WITH B700<7700.7FFFM.

6. MOVE DEMUFFIN AS BEFORE WITH 803<4000.58FFM.

7. TYPE 803G AND PROCEED WITH THE FILE TRANSFER.

THERE ARE A FEW CASES WHERE JUST A LITTLE MORE INTELLIGENCE IS USED TO MODIFY DOS AFTER THE PROGRAM BEGINS TO RUN (USUALLY THE MODIFICATION OF PROLOG/EPILOG BYTES, BUT SOMETIMES A LITTLE BIT MORE), AND IN THIS CASE YOU HAVE TO BOOT THE DISK AND LET IT RUN A SECOND OR TWO BEFORE RESETTING. THE INSPECTOR IN ROM IS A BIG HELP IN A CASE LIKE THIS: YOU CAN RESET AND SCAN THE ENTIRE DISK WITH THE

'SHIFT +' KEYS TO SEE IF ALL THE SECTORS CAN BE READ WITH THE DOS CURRENTLY IN THE SYSTEM. IF THEY CAN BE, CHANCES ARE GOOD THAT YOU WILL BE SUCCESSFUL WITH THE CONVERSION. AFTER RESETTING, SAVE THE ENTIRE DOS WITH D00<9D00.BFFFM, THEN BOOT UP YOUR STANDARD DISK AND BLOAD DEMUFFIN PLUS. PUT THE MODIFIED DOS BACK WITH 9D00<D00.2FFFM, AND PROCEED WITH STEPS 6 AND 7 ABOVE.

  ASSUMING THAT THE ORIGINAL COPY WAS GOOD, AND THAT NO SECONDARY PROTECTION WAS USED, YOU SHOULD NOW HAVE A COPYA VERSION OF THE PROGRAM. IN MANY CASES, IT'S POSSIBLE TO DO THE JOB WITH EVEN LESS HASSLE THAN THIS, SO LET'S LOOK AT WHAT IS (MAYBE) AN EVEN EASIER WAY.

  MANY OF THE RWTS MODIFICATIONS ARE FAIRLY TRIVIAL, AND THE MOST COMMON CONSIST ONLY OF CHANGING THE PROLOG OR EPILOG BYTES FOR THE ADDRESS OR DATA FIELD. YOU CAN OFTEN PRODUCE AN UNPROTECTED VERSION OF THESE DISKS BY MAKING A FEW-BYTE CHANGE TO THE RWTS IN MEMORY, AND THEN RUNNING COPYA. THE FOLLOWING DISASSEMBLY CONTAINS THE ROUTINES WHICH READ IN THE ADDRESS AND DATA FIELDS, AND WHICH NEED TO BE MODIFIED TO CIRCUMVENT A LARGE NUMBER OF RWTS CHANGE SCHEMES:

```
B8DC- A0 20     LDY   #$20
B8DE- 88        DEY
B8DF- F0 61     BEQ   $B942
B8E1- BD 8C C0    LDA   $C08C,X
B8E4- 10 FB     BPL   $B8E1
B8E6- 49 D5     EOR   #$D5
B8E8- D0 F4     BNE   $B8DE
B8EA- EA        NOP
B8EB- BD 8C C0    LDA   $C08C,X
B8EE- 10 FB     BPL   $B8EB
B8F0- C9 AA     CMP   #$AA
B8F2- D0 F2     BNE   $B8E6
B8F4- A0 56     LDY   #$56
B8F6- BD 8C C0    LDA   $C08C,X
B8F9- 10 FB     BPL   $B8F6
B8FB- C9 AD     CMP   #$AD
B8FD- D0 E7     BNE   $B8E6

B8FF- A9 00     LDA   #$00
B901- 88        DEY
B902- 84 26     STY   $26
B904- BC 8C C0    LDY   $C08C,X
B907- 10 FB     BPL   $B904
B909- 59 00 BA    EOR   $BA00,Y
B90C- A4 26     LDY   $26
B90E- 99 00 BC    STA   $BC00,Y
B911- D0 EE     BNE   $B901
B913- 84 26     STY   $26
B915- BC 8C C0    LDY   $C08C,X
B918- 10 FB     BPL   $B915
B91A- 59 00 BA    EOR   $BA00,Y
B91D- A4 26     LDY   $26
B91F- 99 00 BB    STA   $BB00,Y
B922- C8        INY
B923- D0 EE     BNE   $B913

B925- BC 8C C0    LDY   $C08C,X
1928- 10 FB     BPL   $B925
B92A- D9 00 BA    CMP   $BA00,Y
B92D- D0 13     BNE   $B942
```

```
B92F- BD 8C C0    LDA   $C08C,X
B932- 10 FB     BPL   $B92F
B934- C9 DE     CMP   #$DE
B936- D0 0A     BNE   $B942
B938- EA        NOP
B939- BD 8C C0    LDA   $C08C,X
B93C- 10 FB     BPL   $B939
B93E- C9 AA     CMP   #$AA
B940- F0 5C     BEQ   $B99E
B942- 38        SEC
B943- 60        RTS
B944- A0 FC     LDY   #$FC
B946- 84 26     STY   $26
B948- C8        INY
B949- D0 04     BNE   $B94F
B94B- E6 26     INC   $26
B94D- F0 F3     BEQ   $B942

B94F- BD 8C C0    LDA   $C08C,X
B952- 10 FB     BPL   $B94F
B954- C9 D5     CMP   #$D5
B956- D0 F0     BNE   $B948
B959- BD 8C C0    LDA   $C08C,X
B95C- 10 FB     BPL   $B959
B95E- C9 AA     CMP   #$AA
B960- D0 F2     BNE   $B954
B962- A0 03     LDY   #$037
B964- BD 8C C0    LDA   $C08C,X
B967- 10 FB     BPL   $B964
B969- C9 96     CMP   #$96
B96B- D0 E7     BNE   $B954

B96D- A9 00     LDA   #$00
B96F- 85 27     STA   $27
B971- BD 8C C0    LDA   $C08C,X
B974- 10 FB     BPL   $B971
B976- 2A        ROL
B977- 85 26     STA   $26
B979- BD 8C C0    LDA   $C08C,X
B97C- 10 FB     BPL   $B979
B97E- 25 26     AND   $26
B980- 99 2C 00    STA   $002C,Y
B983- 45 27     EOR   $27
B985- 88        DEY
B986- 10 E7     BPL   $B96F
B988- A8        TAY
B989- D0 B7     BNE   $B942

B98B- BD 8C C0    LDA   $C08C,X
B98E- 10 FB     BPL   $B98B
B990- C9 DE     CMP   #$DE
B992- D0 AE     BNE   $B942
B994- EA        NOP
B995- BD 8C C0    LDA   $C08C,X
B998- 10 FB     BPL   $B995
B99A- C9 AA     CMP   #$AA
B99C- D0 A4     BNE   $B942
B99E- 18        CLC
```

```
B99F- 60        RTS
```

BEFORE WE GET INTO ALTERATIONS OF THIS CODE, LET'S GET FAMILIAR WITH THE TERRAIN. THERE ARE TWO SUBROUTINES: 'READ', WHICH READS IN A SECTOR OF DATA AND LIVES FROM $B8DC TO $B943; AND 'RDADR', WHICH READS IN THE ADDRESS FIELD FOR A SECTOR FROM $B944 TO B99F. NOTE THAT THESE ARE IN THE REVERSE ORDER OF THEIR USE IN READING A SECTOR. LET'S LOOK FIRST AT RDADR: AFTER SETTING UP SOME PRELIMINARIES AT $B944-$B94E, WE BEGIN TO LOOK ($B94F-$B96C) FOR THE THREE FAMOUS BYTES OF D5 AA 96 TO IDENTIFY THE START OF THE FIELD. AFTER THEY ARE FOUND, THE VOLUME NUMBER, TRACK NUMBER, AND SECTOR NUMBER ARE STORED IN LOCATIONS $2F, $2E, AND $2D, RESPECTIVELY, AND THE CHECKSUM FOR THE ADDRESS FIELD IS VERIFIED ($B96D-$B98A). FINALLY, THE TWO EPILOG BYTES OF DE AND AA ARE SOUGHT AT THE END OF THE FIELD ($B98B- $B99F).

AFTER AN ADDRESS FIELD IS SUCCESSFULLY READ, 'READ' IS EXECUTED TO READ IN THE DATA FIELD. THE CODE FROM $B8DC TO $B8FE FINDS THE HEADER BYTES OF D5 AA AD, AND THE DATA SECTOR IS READ INTO A PAIR OF BUFFERS WITH THE CODE AT $B8FF-B924 (THE "NIBBLIZING" PROCESS STORED THE 256 BYTES FROM A PAGE OF MEMORY AS A TOTAL OF 342 "NIBBLES" IN THE SECTOR, BUT LET'S NOT GET TOO WORRIED ABOUT THAT YET). FINALLY, THE CHECKSUM (ONE BYTE) IS CHECKED, AND THE EPILOG BYTES ARE ONCE AGAIN VERIFIED ($B925-$B941). NOTICE THE INNOCENT-APPEARING "SEC RTS" AT B942-B943. THIS IS THE HEART OF THE ERROR-DETECTION PROCESS, AND MOST FREQUENTLY MODIFIED (FOR OUR PURPOSES) PART OF THE ENTIRE ROUTINE.

THE ONE BYTE WHICH YOU SHOULD BECOME MOST FAMILIAR WITH IN ORDER TO DO ANY KRACKING, SNOOPING, OR DISK REPAIR IS THE $38 AT LOCATION $B942. THE CARRY BIT (OF THE PROCESSOR STATUS WORD) IS USED THROUGHOUT THE RWTS ROUTINES TO INDICATE A DISK I/O ERROR. WHENEVER ANYTHING GOES WRONG, THE ROUTINES BRANCH TO $B942 TO SET THE CARRY AND RETURN. THE OTHER ROUTINES IN RWTS MONITOR THE CARRY BIT, AND CHECK IT TO SEE IF THERE WAS A BAD ADDRESS READ, A BAD DATA READ, NO HEADER BYTES, WRONG EPILOG BYTES, ETC., ETC.

```
 => THE MOST IMPORTANT CHANGE YOU <=
 => CAN LEARN TO MAKE IS CHANGING <=
 => $B942 TO $18 (OR, IF YOU ARE  <=
 => HOPELESSLY BASIC-BOUND, POKE  <=
 => 47426,24).                  <=
```

THE $18 IS 'CLC' OR 'CLEAR THE CARRY'. BY CHANGING IT, YOU ARE SAYING TO THE RWTS ROUTINES: "DON'T EVEN LOOK TO SEE IF THERE WERE ANY ERRORS. ASSUME EVERYTHING IS ALL RIGHT AND GO ON". THIS IS OBVIOUSLY NOT A GOOD GENERAL PROGRAMMING PRACTICE, SINCE YOU'RE DEFEATING ALL OF THE CAREFUL ERROR- CHECKING THAT DOS DOES, BUT IT'S VERY HANDY TO ALLOW COPYING OF A MODIFIED DOS. IT WILL GENERALLY HANDLE CHANGES IN THE EPILOG BYTES OR INTENTIONAL ERRORS IN THE CHECKSUM OF EITHER FIELD, BUT NOT IN THE HEADER BYTES. HEADER CHANGES (BECAUSE
THE BYTES ARE INDIVIDUALLY CHECKED FOR) MUST BE DONE BY MODIFYING THE APPROPRIATE CODE IN THE SUBROUTINE. IN MANY CASES, THIS IS THE ONLY CHANGE WHICH WILL BE REQUIRED TO MAKE A COPYA VERSION OF THE DISK.

FOR INSTANCE, LET'S SUPPOSE YOU ARE TRYING TO KRACK A PROGRAM, AND YOU SUSPECT THAT THE PROTECTION CONSISTS OF A MODIFIED DOS. READ IN AN ENTIRE TRACK WITH THE INSPECTOR OR NIBBLES AWAY II (THERE IS A BUG IN THE SHIFT-N COMMAND IN SOME VERSIONS OF THE INSPECTOR--YOU CAN'T DO A NIBBLE READ ON ANOTHER TRACK UNLESS YOU FIRST NIBBLE-READ IN TRACK ZERO). EXAMINE AN ADDRESS FIELD AND ITS DATA FIELD. IF YOU FIND BOTH 'D5 AA 96' AND 'D5 AA AD', THEN REMOVE THE DISK AND BOOT UP COPYA. WHILE THE PROGRAM IS ASKING FOR THE SLOT AND DRIVE INFORMATION, PRESS RESET OR TYPE CTRL-C. DELETE LINE 70 (LINE 90 IF YOU ARE USING THE

INTEGER VERSION CALLED "COPY"), THEN FROM THE MONITOR CHANGE

```
         *B942:18
```

  RE-ENTER BASIC AND RUN THE PROGRAM. CHANCES ARE VERY GOOD THAT THE RESULT WILL BE A COPYA VERSION OF THE DISK. BE AWARE, HOWEVER, THAT YOU CAN PROPAGATE OR GENERATE ERRORS IN THIS PROCESS, SINCE ALL OF THE ERROR-CHECKING IN RWTS HAS BEEN TURNED OFF. AS ALWAYS, CHECK THE PROGRAM OUT THOROUGHLY AFTER KRACKING.

  IF YOUR EARLIER SNOOPING REVEALED NON-STANDARD HEADER BYTES, MAKE THE CHANGES LISTED BELOW AFTER RUNNING AND INTERRUPTING COPYA:

```
  ADDRESS FIELD: $B955 - BYTE #1
              B95F - BYTE #2
              B96A - BYTE #3

     DATA FIELD:  B8E7 - BYTE #1
              B8F1 - BYTE #2
              B8FC - BYTE #3
```

  THEN PROCEED AS DESCRIBED EARLIER.

  REGARDLESS OF WHETHER YOU MAKE THESE SIMPLE MODS, OR GO THROUGH THE DEMUFFIN PLUS PROCESS, BEAR IN MIND THAT SECONDARY PROTECTION SCHEMES CAN DEFEAT THESE ATTEMPTS AND REQUIRE YOU TO DO MUCH MORE IN THE WAY OF SNOOPING AND UNDOING. WE'LL PICK UP WITH A DISCUSSION OF THOSE TECHNIQUES NEXT TIME, AND PERHAPS BEGIN TO EXPLORE SOME NON-STANDARD DISK FORMATS.

  OUR QUOTATION OF THE WEEK (MONTH?) IS FROM DON LANCASTER, IN THE INTRODUCTION TO HIS BOOK "ENHANCING YOUR APPLE II, VOL. 1" (A SEMI-GOOD BUT SERIOUSLY "STRETCHED" COMPILATION OF LITTLE HARDWARE TRICKS TO MAKE YOUR APPLE DO NEW THINGS):

  "ANY ATTEMPT AT COPY PROTECTION WILL HACK OFF AND INCONVENIENCE YOUR LEGITIMATE USERS, AND IT WILL DRAMATICALLY INCREASE THE NUMBER OF BOOTLEG COPIES IN CIRCULATION...

  "THE BIG THING ABOUT COPY PROTECTION IS THAT IT DOESN'T. A YEAR'S EFFORT BY A CRACKERJACK MILITARY CRYPTOGRAPY TEAM CAN USUALLY BE UNDONE IN FIFTEEN MINUTES, BETWEEN KLINGON ZAPPINGS, BY YOUR AVERAGE FOURTEEN- YEAR-OLD. AND, MORALITY AND ECONOMICS ASIDE, ONE FACT STANDS OUT...

  UNDOING COPY PROTECTION IS FUN!

  "NOT ONLY IS IT FUN, BUT CRACKING THE UNCOPYABLE IS ABOUT THE MOST CHALLENGING AND REWARDING THING THAT YOU CAN POSSIBLY DO WITH YOUR APPLE. AND, THE THINGS YOU LEARN ALONG THE WAY ARE EXACTLY THE SKILLS THAT YOU WILL NEED TO BECOME A REALLY GREAT PROGRAMMER. SO, I GUESS WE SHOULD ALL BE THANKFUL FOR THE COPY-PROTECTION PEOPLE SINCE THEY ARE ARE GIVING US ALL THIS FASCINATING ENTERTAINMENT AND SUPERB TRAINING AT AN UNBEATABLE PRICE."

  BEAUTIFULLY PUT, DON; AN EXCELLENT RENDITION OF THE "KRACKIST'S MANIFESTO".

```
---------------------------------------
***************************************
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*                             *
*     THE BASICS OF KRACKING 106      *
*                             *
*  MATING ZONE & NIBBLIZING MYSTERIES *
*                             *
*                             *
***************************************
```

CONGRATULATIONS ARE DUE TO TOM LUHRS AND THE PEOPLE AT DATAMOST, FOR PROVIDING BOTH AN ENJOYABLE GAME AND AN ENJOYABLE CHALLENGE IN KRACKING THEIR LATEST OFFERING: "MATING ZONE". THE GAME IS DEFINITELY ABOVE AVERAGE FOR A SHOOT-EM-UP, WITH A NOVEL CONCEPT AND GOOD VARIETY IN THE BEHAVIOUR OF MATED PAIRS, EXPLODING EGGS, AND MULTIPLE LEVELS. THE KRACK IS A LITTLE MORE DIFFICULT THAN THE MODIFIED DOS'S WE HAVE BEEN DISCUSSING, BUT WE ARE STILL DEALING WITH A RELATIVELY STANDARD RWTS.

AS SUPPLIED, THE GAME CAN BE COPIED WITH NIBBLES AWAY II (NO PARMS NEEDED) FOR TRACKS 0-F AND 10.5 TO 13.5. EXCEPT FOR THE HALF TRACKING, THE ONLY DEVIATION FROM NORMAL DOS 3.3 SECTOR STRUCTURE IS AN EPILOG OF 'DF AA' INSTEAD OF THE NORMAL 'DE AA', FOR BOTH THE ADDRESS AND DATA FIELDS. THE DISK ACCESS IS CONTROLLED BY AN ABBREVIATED RWTS LOADED ACROSS SCREEN MEMORY, WITH A CUTE LITTLE SURPRISE AT THE END. I'LL START WITH A SYNOPSIS OF THE KRACKING PROCESS, AND EXPLAIN THE JUICY PARTS IN DETAIL LATER. THE DESCRIPTION WILL BE LIMITED TO REDUCING THE GAME TO A COPYA DISK; HOWEVER, I STRONGLY SUSPECT THAT THE GAME CAN BE STUFFED INTO A SINGLE LONG BFILE (I KNOW, I THOUGHT THAT SIGMA 7 COULD BE, TOO, BUT THAT'S ANOTHER STORY FOR A LATER TIME).

THE KRACKING SEQUENCE IS TO FIRST RELOCATE THE HALF-TRACKS, THEN ELIMINATE THE SECONDARY PROTECTION. AMONG OTHER UTILITIES, NIBBLES AWAY II CAN BE USED FOR THE MOVE AS FOLLOWS:

1. BOOT NA II AND COPY TRACKS 0-F ONTO AN INITIALIZED DISK (YOU'LL WANT THE OTHER TRACKS COPYABLE LATER).

2. SELECT THE TRACK/BIT EDITOR (T), THEN READ IN TRACK 10.5. TYPE 'Z' TO ALLOW NA II TO ANALYZE THE TRACK FOR WRITE-OUT.

3. CHANGE THE TRACK TO 10 EVEN, THEN INSERT THE COPY DISK. TYPE 'W' TO WRITE, THEN 'Y' TO CONFIRM. WHAT WAS ON TRACK 10.5 OF THE ORIGINAL IS NOW ON TRACK 10 OF YOUR COPY.

4. REPEAT STEPS 2 & 3 FOR TRACKS 11.5, 12.5, AND 13.5.

5. TELL THE PROGRAM THAT THE TRACKS HAVE BEEN CHANGED BY MODIFYING TRACK 1,SECTOR F, BYTE 19 FROM $1F TO $1E.

6. CORRECT THE EPILOG BYTE CHECK BY CHANGING $DF TO $DE IN BYTES 35 AND AB OF T0,S5; AND BYTE 9B OF T0,SD.

7. ELIMINATE THE SECONDARY PROTECTION AND THE HIGH SCORE WRITE TO DISK BY CHANGING THE FOLLOWING BYTES:

```
 T0 S5 BYTES E8-EA CHANGE TO 4C B5 04
 T4 SC  BYTE 38    CHANGE TO 60
```

8. LOAD UP COPYA, DEFEAT THE CHECKSUM BY CHANGING $B942 TO $18, THEN MAKE A COPY OF THE DISK.

  9. BOOT AND ENJOY.

  THAT'S THE PROCEDURE, NOW LET'S GO OVER THE THEORY: NORMALLY, RWTS OCCUPIES THE MEMORY SPACE FROM $B700 TO $BFFF. IN ORDER TO SQUEEZE IT INTO $400-7FF, COMPROMISES MUST BE MADE. FIRST, A MINIUMUM OF ONE COMPLETE TRACK IS READ IN, AND THE SECTORS ARE DESTINED FOR SEQUENTIAL PAGES IN MEMORY, BUT WITHOUT THE INTERLEAVING USED BY DOS 3.3. TRACKS ARE REFERRRED TO BY THE EQUIVALENT NUMBER OF HALF-TRACKS: TRACK 6 IS C, F IS 1E, 10.5 IS 21, ETC. THE TRACK READ ROUTINE INCREMENTS THE TRACK NUMBER BY TWO, THEN READS IN THE 16 SECTORS OF THE NEW TRACK.   EXAMINING THE CODE FROM $4DC-55D SHOWS A NORMAL DATA FIELD READ ROUTINE
WITH STANDARD POST-NIBBLIZING TO RECONSTRUCT THE ORIGINAL BYTES:

```
04DC- A0 20     LDY   #$20
04DE- 88        DEY
04DF- F0 7B     BEQ   $055C
04E1- AD EC C0    LDA   $C0EC
04E4- 10 FB     BPL   $04E1
04E6- 49 D5     EOR   #$D5
04E8- D0 F4     BNE   $04DE
04EA- EA        NOP
24EB- AD EC C0    LDA   $C0EC *
04EE- 10 FB     BPL   $04EB
04F0- C9 AA     CMP   #$AA
04F2- D0 F2     BNE   $04E6
04F4- A0 56     LDY   #$56
04F6- AD EC C0    LDA   $C0EC *
04F9- 10 FB     BPL   $04F6
04FB- C9 AD     CMP   #$AD
04FD- D0 E7     BNE   $04E6
04FF- A9 00     LDA   #$00
0501- 88        DEY
0502- 84 26     STY   $26
0504- AC EC C0    LDY   $C0EC *
0507- 10 FB     BPL   $0504
0509- 59 00 07    EOR   $0700,Y
050C- A4 26     LDY   $26
050E- 99 00 03    STA   $0300,Y
0511- D0 EE     BNE   $0501
0513- 84 26     STY   $26
0515- AC EC C0    LDY   $C0EC *
0518- 10 FB     BPL   $0515
051A- 59 00 07    EOR   $0700,Y
051D- A4 26     LDY   $26
051F- 99 00 02    STA   $0200,Y
0522- C8        INY
0523- D0 EE     BNE   $0513
0525- AC EC C0    LDY   $C0EC *
0528- 10 FB     BPL   $0525
052A- D9 00 07    CMP   $0700,Y
052D- D0 2D     BNE   $055C
052F- AD EC C0    LDA   $C0EC *
0532- 10 FB     BPL   $052F
0534- C9 DF     CMP   #$DF
0536- D0 24     BNE   $055C
0538- EA        NOP
0539- AD EC C0    LDA   $C0EC *
053C- 10 FB     BPL   $0539
```

```
053E- C9 AA     CMP   #$AA
0540- D0 1A     BNE   $055C
0542- A0 00     LDY   #$00
0544- A2 56     LDX   #$56
0546- CA        DEX
0547- 30 FB     BMI   $0544
0549- B9 00 02    LDA   $0200,Y
054C- 5E 00 03    LSR   $0300,X
054F- 2A        ROL
0550- 5E 00 03    LSR   $0300,X
0553- 2A        ROL
0554- 99 00 3F    STA   $3F00,Y
0557- C8        INY
0558- D0 EC     BNE   $0546
055A- 18        CLC
055B- 60        RTS
055C- 38        SEC
055D- 60        RTS
```

  (* = THESE INSTRUCTIONS START OUT AS 'C08C', AND HAVE THE SLOT-DEPENDENT VALUE OF 'C0EC' POKED IN AT RUN-TIME. ACCORDING TO MR. SLIPPERY, "REAL MEN WRITE SELF-MODIFYING CODE!").

  THIS IS AS IT SHOULD BE FOR ALL NORMAL SECTOR READING. AT LOCATION $5E8, HOWEVER, ANOTHER DATA FIELD READ ROUTINE BEGINS. THIS IS VERY SUSPICIOUS, INDEED-- WHY SHOULD THEY WASTE SPACE ON A ->SECOND<- DATA FIELD READ ROUTINE, ESPECIALLY IN A "SQUEEZED" RWTS LIKE THIS (REMEMBER THE THIRD LAW OF KRACKING: ACCEPTANCE OF UNUSUAL CODE IS NO VIRTUE; SUSPICION TO THE POINT OF PARANOIA IS NO VICE):

```
05E5- 20 44 07    JSR   $0744
05E8- 20 B9 04    JSR   $04B9
05EB- A0 20     LDY   #$20
05ED- 88        DEY
05EE- F0 F8     BEQ   $05E8
05F0- AD EC C0    LDA   $C0EC
05F3- 10 FB     BPL   $05F0
05F5- 49 D5     EOR   #$D5
05F7- D0 F4     BNE   $05ED
05F9- EA        NOP
05FA- AD EC C0    LDA   $C0EC
05FD- 10 FB     BPL   $05FA
05FF- C9 AA     CMP   #$AA
0601- D0 F2     BNE   $05F5
0603- EA        NOP
0604- AD EC C0    LDA   $C0EC
0607- 10 FB     BPL   $0604
0609- C9 AD     CMP   #$AD
060B- D0 E8     BNE   $05F5
060D- A2 31     LDX   #$31
060F- EA        NOP
0610- 86 26     STX   $26
0612- AC EC C0    LDY   $C0EC
0615- 10 FB     BPL   $0612
0617- B9 00 07    LDA   $0700,Y
061A- 9D 00 02    STA   $0200,X
061D- EA        NOP
061E- EA        NOP
```

```
061F- EA        NOP
0620- CA        DEX
0621- 10 EF     BPL   $0612
0623- AD EC C0    LDA   $C0EC
0626- 10 FB     BPL   $0623
0628- C9 DF     CMP   #$DF
062A- D0 BC     BNE   $05E8
062C- AD EC C0    LDA   $C0EC
062F- 10 FB     BPL   $062C
0631- 49 AA     EOR   #$AA
0633- D0 B3     BNE   $05E8
0635- A2 31     LDX   #$31
0637- 5D 00 02    EOR   $0200,X
063A- CA        DEX
063B- 10 FA     BPL   $0637
063D- 0A        ASL
063E- D0 A8     BNE   $05E8
0640- 4C B5 04    JMP   $04B5
```

ONCE AGAIN, THE CANONICAL PROLOG BYTES OF 'D5 AA AD' ARE LOCATED, AND THEN, STANGELY, ONLY $31 (49 DECIMAL) NIBBLES INSTEAD OF THE NORMAL $156 (342 DECIMAL) ARE READ IN BEFORE SEARCHING FOR THE EPILOG OF 'DF AA'. THE BYTES READ IN ARE EOR'ED TOGETHER, AND IF THE RESULT, SHIFTED LEFT ONCE, IS NOT ZERO, THE READ IS REDONE. THIS IS A VERY CLEVER LITTLE ANTI-COPY ROUTINE WHICH WORKS AS FOLLOWS: WHEN THE SECTOR IS READ INTO MEMORY FROM THE DISK, ALL $156 (342 DECIMAL) NIBBLES ARE READ IN AND POSTNIBBLIZED TO RECONSTRUCT 256 BYTES. THESE BYTES ARE THEN RE-NIBBLIZED AND WRITTEN OUT TO THE NEW DISK BEING MADE. SINCE THE ORIGINAL NIBBLES OF DF AND AA WERE NOT THE RESULT OF A PRENIBBLIZING PROCESS, THEY WILL HAVE DISAPPEARED AS THE 50TH AND 51ST NIBBLES OF THE DATA FIELD, AND BEEN REPLACED BY THE NIBBLES WHICH RESULT FROM A LEGITIMATE NIBBLIZING AND EXCLUSIVE-ORING PROCESS. ONE PASS THROUGH ANY STANDARD DOS SECTOR-BASED COPIER WILL THUS "DESTROY" THE SECTOR FROM THE STANDPOINT OF THE PROTECTION SCHEME.

THE SCHEME, ALTHOUGH A SUBTLE AND CUTE SECONDARY PROTECTION SYSTEM, IS RATHER HOLLOW, SINCE NOTHING FURTHER IS DONE WITH THE CHECKSUM OF THE $31 NIBBLES. THIS WAY, AS SOON AS THE SCHEME IS DECODED, THE ROUTINE CAN SIMPLY BE BYPASSED WITH NO PENALTY.

NOW, I REALIZE THAT THIS DISCUSSION BROUGHT MANY OF YOU TO THE "MEGO" POINT (GOVERNMENT-TALK ACRONYM FOR "MY EYES GLAZE OVER") AT THE FIRST USE OF THE TERMS PRE- AND POST- NIBBLIZING. THOSE WHO KNOW IT ALL NEED READ NO FURTHER, BUT FOR THOSE TO WHOM THIS IS STILL DIALECTIC SWAHILI, I WILL HUMBLY OFFER MY VERSION OF AN EXPLANATION (IN FULL KNOWLEDGE THAT IT MAY DO NO MORE THAN INCREASE THE EYE-GLAZE COEFFICIENT). ONCE AGAIN, THE PRIMARY SOURCE FOR THIS SORT OF EXPOSITION IS "BENEATH APPLE DOS", WHICH BY NOW HAS ACHIEVED THE STATUS OF THE MOST FREQUENTLY-REFERENCED TEXT IN ALL OF APPLE KRACKDOM, IF NOT APPLE PROGRAMMING IN GENERAL.

WE'RE STILL NOT READY TO GO INTO THE INNERMOST WORKINGS OF THE DISK STORAGE PROCESS (THAT'LL BE BASICS 107) BUT LET'S STIPULATE FOR THE MOMENT THAT THERE IS A NEED TO USE ONLY BYTES WHICH MEET CERTAIN STRICT REQUIREMENTS WHEN WRITING ONTO AN APPLE DOS 3.3 DISK. THE STONE TABLETS CARRIED UP FROM CUPERTINO LIST THOSE REQUIREMENTS:

1. THE HIGH BIT OF THE BYTE MUST BE '1'

2. THE BYTE MUST CONTAIN NO MORE THAN ONE PAIR OF ADJACENT ZEROES.

3. THERE MUST BE AT LEAST TWO ADJACENT ONES IN THE BYTE, NOT INCLUDING THE HIGH BIT.

AS IT TURNS OUT, THERE ARE EXACTLY 64 BYTES WHICH MEET ALL OF THESE CRITERIA. IN ORDER TO STORE INFORMATION ON THE DISK, WE MUST "ENCODE" A TOTAL OF 256 BYTES (ONE PAGE AND ALSO ONE SECTOR) IN THE 64 PSEUDO-BYTES WHICH CAN BE WRITTEN.

64 DIFFERENT BYTES MEANS THAT WE CAN SET UP A TABLE IN WHICH EACH BYTE UNIQUELY CORRESPONDS TO ONE OF THE SIX-BIT NUMBERS FROM 00 TO $3F (IN BINARY, 0000 0000 TO 0011 1111). THE PROCESS OF CHOPPING UP FULL 8-BIT BYTES INTO PIECES WHICH CAN CORRESPOND TO 6-BIT BYTES IS CALLED "NIBBLIZING". WE CAN BEGIN TO SIMULATE THE "PRENIBBLIZING" PROCESS BY MAKING UP TWO TABLES. THE FIRST ONE, WHICH NORMALLY RESIDES AT $BB00-BBFF, IS SET UP TO CONTAIN THE FIRST SIX BITS OF EACH OF THE 256 BYTES:

```
  IF THE      THE CORRESPONDING
ORIGINAL BYTE  ENTRY IN THE BB00
 ALUE WAS:    TABLE IS:
                          TABLE

(HEX) (BINARY)     (HEX)      (BINARY)   ADDR
 00   0000 0000    00   0000 0000  BB00
 3F   0011 1111    3F   0011 1111  BB01
 47   0100 0111    07   0000 0111  BB02
 69   0110 1001    29   0010 1001  BB03
 7F   0111 1111    3F   0011 1111  BB04
 85   1000 0101    05   0000 0101  BB05
 BC   1011 1100    3C   0011 1100  BB06
 F0   1111 0000    30   0011 0000  BB07
 FF   1111 1111    3F   0011 1111  BB08
                    !    !     !
                    !    !     !
                    V    V     V
```

AS YOU CAN SEE, IN EACH CASE THE FIRST TWO BITS HAVE BEEN CHOPPED OFF AND REPLACED WITH ZEROES. THE RESULTING BYTE, NOW BETWEEN 0 AND $3F (0 AND 63) CAN BE RELATED, ONE FOR ONE, TO THE WRITEABLE BYTES. HOWEVER, IF WE DON'T STORE, IN SOME ORGANIZIED FASHION, THOSE TWO BITS WE LOPPED OFF EVERY BYTE, WE WON'T BE ABLE TO RECONSTRUCT THE ORIGINAL BYTES WHEN WE READ THESE FUNNY LITTLE 6-BIT NIBBLEBYTES FROM THE TRACK. THE WAY THAT'S DONE IS TO CONTRUCT A SECOND TABLE, NORMALLY AT BC00-BC55, WHICH CONTAINS ALL THE LITTLE BITS AND PIECES (HO-HO-HO) LEFT OVER AFTER THE TRUNCATION OF THE ORIGINAL BYTES TO SIX BITS. IN THE EXAMPLE LIST GIVEN ABOVE, THE LEFTOVERS ARE:

```
       00 -------------------\
       00 ---------------\    \
       01 -------------\  \    \
       01        \  \  \
       01         \  \  \
       10          \  \        \
       10           !  !   !
       11           !  !   !
       11           !  !   !
                    !  !   !
THIS SECOND LIST WORKS     !  !   !
FROM THE BOTTOM UP, SO     !  !   !
THE CONTENTS WOULD BE:     !  !   !
                    !  !   !
```

```
      ^     ^   ^    !  !   !
      !     !   !    !  !   !
      !     !   !    !  !   !
      BC4D    00XX YY11    !  !   !
      BC4E    00XX YY11    !  !   !
      BC4F    00XX YY10    !  !   !
      BC50    00XX YY10    !  !   !
      BC51    00XX YY01    !  !  /
      BC52    00XX YY01   /  /      /
      BC53    00XX YY01 -/  /  /
      BC54    00XX YY00 ---/  /
      BC55    00XX YY00 -----/
```

SO, THE FIRST TWO BITS OF THE ORIGINAL BYTE BECOME THE LAST TWO BITS OF THE BYTES IN THIS TABLE, WORKING FROM THE BOTTOM UP. AFTER $56 (86 DECIMAL) BYTES HAVE HAD THEIR FIRST TWO BITS STUFFED INTO THE TABLE, THE NEXT ONE REPLACES THE "YY" AT LOCATION BC55, THEN AT BC54, ETC. AFTER $AC (172) BYTES, THE NEXT PAIR OF LEFTOVERS GOES INTO THE "XX" SLOT OF LOCATION BC55, AND WORKS UP AGAIN UNTIL THE LAST TWO BITS ARE STUFFED INTO THE "XX" SLOT OF LOCATION BC00. REMEMBER THAT THE TWO MOST SIGNIFICANT BITS MUST ALWAYS BE ZERO TO STAY WITHIN THE 0-3F RESTRICTION.

AFTER THESE TWO TABLES HAVE BEEN CONSTRUCTED, EACH VALUE IN THE TABLE IS EXCLUSIVE-ORED ON WITH THOSE THAT WENT BEFORE, TO FORM A NEW SIX-BIT BYTE. THE RESULTING VALUE, WHICH IS STILL BETWEEN 0 AND $3F, IS TRANSLATED TO ONE OF THE 64 BYTES WHICH OBEY ALL THE LAWS LISTED ABOVE FOR THE DISK BYTES, AND THEN REALLY AND TRULY WRITTEN TO DISK. THE TABLE WHICH DOES THIS CONVERSION IS CALLED THE "WRITE TRANSLATE TABLE" AND LIVES AT $BA29 TO $BA68. IN OUR EXAMPLE, THEN THE PROCESS GOES LIKE THIS:

1. GET A BYTE FROM BB00 =00
2. EXCLUSIVE-OR IT WITH 00 (IT'S THE FIRST BYTE) 00 EOR 00=00
3. LOOK UP THE BYTE AT $BA29 + 0 =96
4. WRITE IT TO DISK.

----THE NEXT BYTE-------------

1. GET THE BYTE FROM BB01 =3F
2. EXCLUSIVE-OR IT WITH THE PREVIOUS VALUE OF 00 3F EOR 00=3F
3. LOOK UP THE BYTE AT $BA29 + $3F FF
4. WRITE IT TO DISK

----AND THE THIRD BYTE--------

1. GET THE BYTE FROM BB02 =07
2. EXCLUSIVE-OR IT WITH THE PREVIOUS VALUE OF 3F 07 EOR 3F=34
3. LOOK UP THE BYTE AT $BA29 + $34 F3
4. WRITE IT TO DISK

----FOURTH AND LAST EXAMPLE---

1. GET THE BYTE FROM BB03 =29
2. EXCLUSIVE-OR IT WITH THE PREVIOUS VALUE OF 34 29 EOR 34=1D
3. LOOK UP THE BYTE AT $BA29 + $1D CE
4. WRITE IT TO DISK

AND SO ON UNTIL THE TOTAL OF $156 OR 342 BYTES FROM THE TWO TABLES IS WRITTEN TO DISK. (LOOKING AT THIS PROCESS, YOU CAN SEE THAT A SECTOR WITH ALL ZEROES

WOULD NEVER CHANGE THE FIRST BYTE WRITTEN OUT, AND WOULD DISPLAY A SECTOR FULL OF 96'S ON A NIBBLE READ).

WHEN THE DATA FIELD OF A SECTOR IS READ BACK IN, THE PROCESS IS REVERSED. AFTER ALL 342 BYTES ARE READ INTO BB00-BC55, EACH BYTE IS EXCLUSIVE-ORED OFF THE PILE, AND THE RESULT IS USED TO LOOK UP A VALUE OF 0-3F IN A "READ TRANSLATE TABLE" AT BA96-BAFF. THROUGH SOME ELEGANT, IF INTRICATE CODE, THIS 6-BIT "BYTE" IS RECOMBINED WITH ITS LONG-LOST 2 BITS, AND THE FINAL, REAL BYTE IS STORED WHERE $3E AND $3F ARE POINTING.

IN THE FEW REMAINING LINES, LET ME EXPOUND FOR A MOMENT ON THE EXCLUSIVE- OR OPERATOR. THE INSTRUCTION EOR (WHICH USES THE MNEMONIC XOR IN EVERY OTHER ASSEMBLY LANGUAGE) WORKS LIKE THIS: FOR EACH BIT OF THE TWO BYTES TO BE OPERATED ON, THE OUTPUT IS A ONE IF ONE AND ONLY ONE OF THE BITS IS ONE, BUT A ZERO IF BOTH ARE ZERO OR ONE. THE TRUTH TABLE BELOW SUMMARIZES:

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

AND THE LAST EXAMPLE GIVEN ABOVE IS:

```
        0010 1001  (29)
    EOR   0011 0100  (34)
        ---------
        0001 1101  (1D)
```

THIS IS BASICALLY A NEAT LITTLE OPERATOR WHICH HAS BEEN FREQUENTLY PERVERTED BY THE ENEMY TO DO THEIR DIRTY WORK. MORE ON THIS LATER--STAY TUNED FOR THE NEXT EPISODE: "NON-STANDARD ENCODING SCHEMES."

```
---------------------------------------
***************************************
*                              *
*                              *
*     KRAKOWICZ'S KRACKING KORNER     *
*                              *
*     THE BASICS OF KRACKING 107      *
*                              *
*    BEYOND DEMUFFIN:NON-STANDARD     *
*      DISK ENCODING TECHNIQUES       *
*      AND DISKBIT TIDBITS          *
*                              *
*                              *
***************************************
```

RECENTLY, WE HAVE DISCUSSED THE UNPROTECTION OF DISKS WITH A MODIFIED DOS, PRIMARILY THROUGH THE USE OF DEMUFFIN PLUS. THIS TIME WE'LL TALK A LITTLE ABOUT SOME OTHER APPROACHES TO CONVERTING MODIFIED DOS DISKS, THEN GET INTO SOME SLIGHTLY HEAVIER STUFF ABOUT THOSE PROTECTION TECHNIQUES WHICH GO WAY BEYOND MODIFYING DOS.

IN ADDITION TO DEMUFFIN PLUS, TWO PROGRAMS HAVE RECENTLY BECOME AVAILABLE FOR UNPROTECTING A DISK WITH MODIFIED DOS: COPYB AND ADVANCED DEMUFFIN. SINCE, AS WE DISCUSSED EARLIER, MOST MODS ARE MADE TO THE RWTS PORTIONS OF DOS, ALL THAT'S REALLY NECESSARY TO REMOVE THE PRIMARY PROTECTION IS TO READ THE DISK INTO

MEMORY USING ITS OWN RWTS, THEN SWITCH IN A NORMAL RWTS AND WRITE IT BACK OUT TO ANOTHER DISK. SINCE THE FILE MANAGER IS NOT INVOKED AS IT IS IN DEMUFFIN PLUS, THIS APPROACH HAS THE ADDED ADVANTAGE THAT A DISK WITH FAIRLY NORMAL SECTOR STRUCTURE BUT NO DOS FILE STRUCTURE OR CATALOG CAN STILL BE CONVERTED. IN THE EARLY DAYS, THE TECHNIQUE WAS THIS: STORE BOTH RWTS'S IN MEMORY, THEN USE THE INSPECTOR TO READ IN ABOUT 8 TRACKS. MOVE THE NORMAL RWTS IN WITH THE MONITOR, THEN WRITE OUT THE TRACKS TO AN INITIALIZED DISK. ABOUT 5 TIMES AROUND MAKES A COPYA DISK AND JACK A DULL BOY. FORTUNATELY, THINGS HAVE IMPROVED.

COPYB IS A MODIFICATION OF COPYA WHICH AUTOMATICALLY SWAPS THE RWTS ROUTINES FOR YOU. TO RUN IT, YOU BOOT THE PROTECTED PROGRAM AND INTERRUPT IT, THEN MOVE THE RWTS ROUTINES FROM $B700 TO $8000 (YOU CAN ALSO USE THE READ-IN TECHNIQUE DESCRIBED IN BASICS 105 TO PUT THE MODIFIED RWTS INTO MEMORY, AND YOU CAN KEEP A LIBRARY OF RWTS'S IF YOU FIND PEOPLE LIKE MUSE AND SSI USING A PARTICULAR ONE OVER AND OVER. THESE CAN SIMPLY BE LOADED AS BFILES INTO $8000 AFTER BOOTING COPYB). BOOTING COPYB AND ANSWERING ONE CRYPTIC AND FOUR FIDDISH QUESTIONS ALLOWS YOU TO PRODUCE A COPYA DISK, INCLUDING AUTOMATIC INITIALIZATION OF THE TARGET DISK. REASONABLY COMPLETE INSTRUCTIONS, WRITTEN BY THE AUTHOR WHO HAS ACHIEVED NATIONAL PROMINENCE FOR VERBAL DIARRHEA, ACCOMPANY THE PROGRAM, SO WE WON'T BELABOR THEM HERE.

A MUCH MORE COMPLETE PROGRAM CALLED ADVANCED DEMUFFIN HAS RECENTLY ISSUED FROM CORRUPT COMPUTING, UNDER THE ABLE AUTHORSHIP OF "THE STACK" AND "THE INSPECTOR". IT ALSO MAKES UNPROTECTED COPIES VIA RWTS SWAPS, BUT IS MUCH MORE USER-FRIENDLY AND VERSATILE. THOROUGH SOFTDOC ACCOMPANIES THAT PROGRAM AS WELL, SO WE NEEDN'T DWELL ON IT, EXCEPT TO CONGRATULATE THE AUTHORS ON AN EXCELLENT AND HIGHLY PROFESSIONAL CONTRIBUTION TO THE ART OF UNPROTECTION.

EXCEPT FOR SOME CLEVER AND WELL-HIDDEN SECONDARY PROTECTION, THERE IS NOT MUCH THAT A PROTECTOR CAN DO THESE DAYS WITH A MODIFIED DOS THAT WE CAN'T UNDO IN SHORT ORDER WITH THE TOOLS AND TECHNIQUES AVAILABLE TO US. WHY ARE THERE STILL SOME PROGRAMS THAT TAKE A LONG TIME TO KRACK? MORE EXTENSIVE MODIFICATIONS, EXTENDING EVEN TO COMPLETE CUSTOM DOS'S. HERE, HOWEVER, WE START TO SEPARATE THE MEN FROM THE BOYS, SINCE WRITING YOUR OWN OPERATING SYSTEM, NO MATTER HOW LIMITED, COSTS MONEY. WHILE DOS MODIFICATIONS, EVEN WITH SEVERAL VARIATIONS, CAN BE WHIPPED OUT IN A FEW MINUTES BY ANY KLUTZY HACKER, READING AND WRITING IN WAYS NOT SANCTIONED BY THE GODS OF APPLEDOS REQUIRE HIRING SOMEONE WHO KNOWS HIS SHIT, AND HE ALSO USUALLY KNOWS HOW MUCH HE'S WORTH. THIS HAS ADVANTAGES FOR US AS WELL, SINCE PUBLISHERS WILL TRY TO GET THEIR MONEY'S WORTH OUT OF AN EXPENSIVE SYSTEM BY USING IT ON AS MANY PRODUCTS AS POSSIBLE. ONCE BROKEN, THE PRICIPLES CAN BE READILY APPLIED TO ALL DISKS OF THE SAME GENERATION OF PROTECTION.

IF A PUBLISHER IS GOING TO GO BEYOND MODIFYING DOS, HE WILL NORMALLY ALSO ABANDON STANDARD TRACK AND SECTOR FORMAT FOR SOMETHING WHICH AFFORDS GREATER SECURITY AND EASE OF USE (SOMETIMES, SINCE THE APPLE DISK HARDWARE IF SO FLEXIBLE, FORMATS WHICH WERE BORN ON ENTIRELY DIFFERENT SYSTEMS FIND THEIR WAY INTO APPLE PROTECTION SCHEMES). GAMES, ESPECIALLY, HAVE MUCH SIMPLER STRUCTURE, AND ARE READILY ADAPTED TO A FORMAT WITH LESS COMPLEXITY. SINCE SPACE ON A GAME DISK IS USUALLY NOT AT A PREMIUM, A VERY COMMON SIMPLIFICATION IS TO ELIMINATE SECTORING ALTOGETHER, AND MAKE EACH TRACK ONE BIG SECTOR. THIS NOT ONLY SIMPLIFIES THE PROGRAM THAT HAS TO READ THE DISK, BUT CAN ALSO DRAMATICALLY INCREASE THE DATA TRANSFER RATE (SIRIUS'S HADRON BROUGHT IN A FULL 48K IN JUST OVER FOUR SECONDS--EAT YOUR HEART OUT, DOS). BEFORE WE DISCUSS SOME OF THE FORMATS USED, WE HAVE TO TAKE A MUCH CLOSER LOOK AT THE WAY INFORMATION IS ACTUALLY READ FROM A DISK.

THERE ARE A FEW ABSOLUTE LAWS OF DISK WRITING AND READING WHICH MUST BE

OBSERVED, AND SEVERAL MINOR STATUTES WHICH MAY BE VIOLATED WITH ONLY A SUMMONS. THE REAL, DEEP DOWN, TRUE WAY THAT DATA IS RECORDED ON ANY DISK IS BY WAY OF "MAGNETIC FLUX CHANGES", THAT IS, REVERSALS IN THE DIRECTION OF MAGNETIZATION OF A THIN COATING OF IRON OXIDE ON THE DISK SURFACE. WE ALL RECALL FONDLY THE SCIENCE EXPERIMENTS WITH IRON FILINGS AND A BAR MAGNET; DISK RECORDING TECHNOLOGY IS BASED ON MAKING THE PARTICLES VERY SMALL, AND IMMOBILIZING THEM ON THE DISK SO THEY CAN BE EXAMINED LATER FOR THE STATE OF THEIR MAGNETIZATION. DISKETTE READING IS ACTUALLY A (GASP!) ANALOG PROCESS, AND IS MADE DIGITAL BY SOME CLEVER CIRCUITRY JUST DOWNSTREAM OF THE READ HEAD. THIS CIRCUITRY SENSES THE MAGNETIC FIELD OVER A PRECISELY DEFINED TIME INTERVAL, AND TRANSLATES A *CHANGE* (REVERSAL) IN THE DIRECTION OF MAGNETIZATION TO A DIGITAL "ONE", AND INTERPRETS *NO CHANGE*, OR THE ABSENCE OF REVERSAL, AS A "ZERO".

"NIBBLE" (IN CASE YOU WERE CURIOUS, THE DISK SPINS AT 300 RPM WHICH IS 5 REVOLUTIONS PER SECOND, OR 200 MILLISECONDS (MSEC) PER ROTATION. SINCE 8 BITS = ONE BYTE, A BYTE IS READ EVERY 32 USEC, OR 0.032 MSEC, AND EACH REVOLUTION OF THE DISK CORRESPONDS TO 200 MSEC/0.032 MSEC OR ABOUT 6000 BYTES. THIS IS ROUGHLY $1800 BYTES PER TRACK, WHICH IS ABOUT THE NUMBER OF BYTES YOU NORMALLY SEE DISPLAYED DURING A NIBBLE COUNT WITH NIBBLES AWAY OR LOCKSMITH).

IT'S NOT TOO BAD A PHYSICAL PICTURE TO REPRESENT THE ORIENTATION OF THE MAGNETIC FIELDS WITH ARROWS (UP AND DOWN ARROWS WOULD BE NICER, BUT THE APPLE SCREEN NO GOTS). IN THE DIAGRAM BELOW, THE ORIENTATION OF MAGNETIC "DOMAINS" ON THE DISK FOR 9 BITS ARE REPRESENTED:

```
READ PT     1  2  3  4  5  6  7  8  9
DIR'N     -> -> <- -> -> -> <- <- -> ->
         \ /\ /\ /\ /\ /\ /\ /\ /\ /
BIT VALUE   0  1  1  0  0  1  0  1  0
```

NOTICE THAT EACH TIME THE MAGNETIC FIELD REVERSES DURING THE READ INTERVAL, THE BIT VALUE IS READ AS "1", AND AS "0" WITH NO REVERSAL.

THE DISK ANALOG CARD AND CONTROLLER CARD COOPERATE TO STACK UP THIS "SERIAL BIT STREAM" INTO AN 8-BIT BYTE, USING A SHIFT REGISTER WHICH IS THE HARDWARE EQUIVALENT OF THE "ASL" OR "ARITHMETIC SHIFT LEFT" INSTRUCTION IN APPLE ASSEMBLY LANGUAGE. THE SHIFT REGISTER STARTS OUT FULL OF ZEROES, AND KEEPS SCHLEPPING IN, FROM THE LEFT, THE NEW BIT READ FROM THE DISK EVERY 4 MICROSECONDS. THE SEQUENCE BELOW REPRESENTS THE SHIFT REGISTER CONTENTS AT EACH OF THE READ POINTS SHOWN IN THE ARROW CHART ABOVE:

```
SHIFT                   NEXT BIT
REGISTER                 TO BE
BIT #  --> 7 6 5 4 3 2 1 0     ADDED
        +---------------+     /
(# OF   !           !    /
 SHIFTS) 0!0 0 0 0 0 0 0 0!<- 0
        !           !
       1!0 0 0 0 0 0 0 0!<- 1
        !           !
       2!0 0 0 0 0 0 0 1!<- 1
        !           !
       3!0 0 0 0 0 0 1 1!<- 0
        !           !
       4!0 0 0 0 0 1 1 0!<- 0
        !           !
       5!0 0 0 0 1 1 0 0!<- 1
        !           !
       6!0 0 0 1 1 0 0 1!<- 0
        !               !
       7!0 0 1 1 0 0 1 0!<- 1
        !               !
       8!0 1 1 0 0 1 0 1!<- 0
        !               !
       9!1 1 0 0 1 0 1 0!
        +---------------+
```

NOTICE THAT THE MOST SIGNIFICANT BIT ("MSB", OR BIT 7) OF THE SHIFT REGISTER STAYS AT "0" UNTIL THE NINTH SHIFT, WHEN A "1" IS SHIFTED IN. THIS IS THE SIGNAL WE USE TO DECIDE WHEN WE SHOULD STOP READING AND SHIFTING, AND CALL IT A BYTE. THE SHIFT REGISTER IS DECODED AS ADDRESS $C0EC (FOR SLOT SIX), AND THE FAMILIAR INSTRUCTION SEQUENCE:

```
  $B954  LDA $C08C,X (X=60 FOR SLOT 6)
         BPL $B954
```

IS USED AS A "WAIT AND WATCH" LOOP TO DETECT WHEN THE MSB HAS FINALLY BECOME A ONE. IF YOU ARE STILL FOLLOWING THE DISCUSSION, YOU SHOULD NOW BE ABLE TO SEE THE REASON FOR THE FIRST LAW OF DISK BYTES (LISTED IN BASICS 106): IF THE FIRST BIT OF THE BYTE WEREN'T A ONE, BIT 7 OF THE SHIFT REGISTER WOULD STILL HAVE A ZERO WHEN WE SHOULD BE AT THE END, AND WE WOULD SHIFT AT LEAST ONE MORE TIME, LOOKING IN VAIN FOR A "1". THE SECOND (NOT MORE THAT ONE PAIR OF ADJACENT ZEROES) IS REQUIRED TO KEEP THE CIRCUITRY FROM GETTING LOST (THE THIRD LAW, WHICH REQUIRES AT LEAST ONE PAIR OF ADJACENT ONES NOT INVOLVING BIT 7, IS ONLY FOR DOS 3.3, AND DOES NOT AFFECT THE HARDWARE). LET'S LOOK, FOR REVIEW, AT SOME LEGAL AND ILLEGAL NIBBLES:

BYTE	BINARY	LEGAL	VIOLATION
7F	0111 1111	NO	RULE 1
8F	1000 1111	NO	RULE 2
92	1001 0010	NO	RULE 2
95	1001 0101	YES	NOT DOS 3.3
96	1001 0110	YES	NONE
97	1001 0111	YES	NONE
98	1001 1000	NO	RULE 2
9A	1001 1010	YES	NOT DOS 3.3
9B	1001 1011	YES	NONE
D5	1101 0101	YES	*
AA	1010 1010	YES	*

*THESE TWO BYTES ARE NOT ALLOWED IN THE DOS 3.3 NIBBLIZING SCHEME, BUT ARE USED IN PROLOGS AND EPILOGS.

ALSO, TUCK THIS AWAY IN THE BACK OF YOUR MIND: *N0* LEGAL DISKBYTES CAN CONTAIN 8, 1, OR 0.

NOW, IF YOU WANT TO CREATE A NON-STANDARD DISK FORMAT TO KEEP THOSE NASTY PIRATES OUT OF YOUR "UNKRACKABLE" SOFTWARE, ALL YOU HAVE TO DO IS PICK A SELECTION OF LEGAL BYTES (AND MAYBE A FEW OF THE ILLEGAL ONES), AND ARRANGE YOUR OWN ENCODING SCHEME. THE MOST COMMON TECHNIQUE IS AN ADAPTATION OF THE OLD ENCODING SCHEME CALLED 4+4 NIBBLIZING INTRODUCED TO DISK PROTECTION (I BELIEVE) BY MY GOOD FRIENDS AT SIRIUS SOFTWARE. THIS IS THE SAME SYSTEM APPLE USES TO STORE VOLUME, TRACK AND SECTOR DATA IN THE ADDRESS FIELD (SEE BASICS 104). AS WE DESCRIBED, EACH REAL BYTE IS SPLIT INTO ODD AND EVEN HALVES, AND ENCODED SO THAT EACH BYTE STORED ON THE DISK REPRESENTS EXACTLY 4 BITS, OR ONE NIBBLE, OF

THE ORIGINAL BYTE (THE BEGINNING OF THE DISKNIBBLEBYTE CONFUSION). THE CHOICE FOR THESE IS LIMITED: ALL DISK BYTES ARE MADE UP OF A,B,E AND F, SO YOU CAN HAVE,ON THE DISK:

```
      AA AB AE AF BB BA BE BF
      EA EB EE EF FA FB FE FF
```

YOU CAN FIND (PERHAPS TOO MUCH) MORE INFORMATION ON THIS TECHNIQUE AND DECODING IT IN THE FILES ON CYCLOD, WAY OUT, AND TYPE ATTACK.

IN GENERAL, WHILE THE APPROACH TO UNPROTECTING ALL OF THESE ODDBALL FORMATS IS STRAIGHTFORWARD, THE WORK CAN BE LONG AND HARD, AND CAN PROVIDE SOME REAL CHALLENGE TO OUR SKILL AS KRACKISTS AND PROGRAMMERS. IN BROAD OUTLINE:

A. TRY TO FIGURE OUT THE DISK ACCESS LOGIC AND ISOLATE THE READER/LOADER CODE.

B. MODIFY IT TO READ IN ALL THE PERTINENT PARTS OF THE DISK.

C. SAVE THE PIECES OUT TO DISK UNDER NORMAL RWTS STRUCTURE.9

D. RECONSTRUCT THE PROGRAM, USING AS LITTLE NEW CODE AS POSSIBLE.

BY NOW YOU SHOULD KNOW WHAT A LOADER ROUTINE LOOKS LIKE, AND IN MOST CASES THESE DISKS WILL LOAD A COMPLETE TRACK AT A TIME INTO A PREDETERMINED AREA OF MEMORY. BY LOCATING AND ALTERING THE TABLE OF "WHERE TO READ IN", YOU CAN, IN A FEW PASSES, READ THE TRACK INTO MEMORY, BOOT A DISK, AND WRITE THE MEMORY CONTENTS OUT UNDER THE NORMAL DOS FORMAT. AFTER ALL THE INFORMATION IS SAVED, YOU CAN BEGIN THE PROCESS OF RECONSTRUCTION. USUALLY, THIS CONSISTS OF LOADING DOS (OR AT LEAST RWTS) INTO MEMORY AND USING IT TO MANIPULATE SECTIONS OF THE ORIGINAL CODE. IN CASES LIKE CYCLOD, THE ADDITIONAL DISK ACCESS BETWEEN LEVELS IS TOTALLY UNNECESSARY AND CAN BE ELIMINATED. IN SOMETHING LIKE BANDITS, HOWEVER, A GREAT DEAL OF REAL DATA IS READ IN AT EACH LEVEL, AND A MEANS MUST BE FOUND TO ACCOMMODATE THE DISK ACCESS. USUALLY, THIS MEANS TRYING TO SQUEEZE THE ABSOLUTELY CRUCIAL SUBROUTINES FROM RWTS INTO THE SPACE ORIGINALLY OCCUPIED BY THE LOADER ROUTINE.

THERE ARE A NUMBER OF THESE "SHORT DOS" ROUTINES IN EXISTENCE. THESE PROGRAMS ARE ALL LESS THAN $400 BYTES LONG, AND INCLUDE TRACK SEEK, ADDRESS AND DATA FIELD READERS, AND POSTNIBBLIZING ROUTINES. IN THE PROCESS OF KRACKING BANDITS, BOTH THE SHY "NAMELESS" KRACKER ("THEY SAID IT COULDN'T BE DONE...") AND I WROTE VIRTUALLY IDENTICAL ROUTINES WHICH LIVED IN TEXT MEMORY AT 400-7FF. LONG- JOHN SILVER HAS HIS OWN VERSION OF A SHORT DOS, AND AN EXCELLENT IMPLEMENTATION HAS RECENTLY BEEN INTRODUCED BY THE STACK AND THE INSPECTOR OF "CORRUPT COMPUTING". IT IS EXTREMELY WELL DOCUMENTED, AND JUST AS IN THE CASE OF ADVANCED DEMUFFIN, THE BEST UTILITY AVAILABLE TO THE PRACTICING KRACKIST.

THAT'S A CRUDE OUTLINE OF THE DISK PROTECTION SCHEMES WHICH GO BEYOND MODIFIED DOS; UNFORTUNATELY, MOST ARE QUITE DIFFERENT IN DETAIL, AND YOUR SKILL AS A KRACKIST MUST BE MADE EQUAL TO THE TASK OF EACH ONE. WE'LL CONTINUE THE BASICS OF KRACKING SERIES NEXT TIME WITH THE LONG-PROMISED ARTICLE ON BOOT- TRACING.

```
---------------------------------------
***************************************
*                             *
*                             *
*     KRAKOWICZ'S KRACKING KORNER     *
*                             *
*                              *
*     THE BASICS OF KRACKING 108:     *
*                              *
*      BOOT CODE TRACING PART 1       *
*                              *
*                              *
*                              *
***************************************
```

AT LAST! THE LONG-AWAITED DESCRIPTION OF BOOT-CODE TRACING AND ITS APPLICATION TO DISK UNPROTECTION. OY KRACKING LAW #7 SAYS "WHEN ALL ELSE FAILS, BOOT TRACE." FOR MANY KRACKISTS, NOTABLE AMONG WHOM WAS MR. XEROX (MAY HE REST IN PEACE), THE MOTTO WAS OPPOSITE: "BEFORE YOU DO ANYTHING ELSE, TRACE THE BOOT CODE." DEPENDING ON YOUR SKILL AND PREDISPOSITION, YOU'LL SETTLE SOMEWHERE IN BETWEEN THESE EXTREMES.

IF MR. XEROX DIDN'T INVENT BOOT- TREACING HE WAS CERTAINLY THE FIRST TO DOCUMENT IT CLEARLY IN THE UNDERGROUND PRESS. THE DESCRIPTION THAT FOLLOWS BORROWS HEAVILY FROM HIS ORIGINAL TREATISE ON THE PIRATE'S HARBOR CRACKING DISO #1. IN ADDMTION, "MYCROFT" WROTE A THOROUGH ARTICLE IN HARDCORE COMPUTING UPDATE 3.1 DESCRIBING HIS OWN, SLIGHTLY DIFFERENT APPROACH TO BOOT-TRACING. WHILE I FIND HIS PROCESS A LITTLE MORE LABORIOUS, IT MIGHT BE NECESSARY FOR SOME VERY DIFFICULT CASES.

THE PROCESS IS BASED FIRMLY ON THE FRIST LAW: TRACK 0, SECTOR 0 OF EVGRY DISK MUST <ALWAYS> LOAD INTO PAGE 8 ($800-8FF). THE FURTHER ASSUMPTION IS THAT, IF WE CAN VIEW EVERY STAGE OF THE BOOT PROCESS, WE CAN LEARN ENOUGH TO PRODUCE AN UNPROTECTED VGRSION OF THE PROGRAM. IT DOES NOT HAVE MYSTICAL POWERS, AND STILL REQUIRES THE ABILITY TO TEAR APART AND UNDERSTAND ASSEMBLY LANGUAGE, MUCH OF WHICH IS INTENTIONALLY MISLEADING. WE'LL BEGIN WITH BACKGROUND MATERIAL AND A REVIEW OF THE NORMAL BOOT PROCESS (DAMMIT, MAUDE, WE ALWAYS HAVE TO SIT THROUGH THE SERMON FIRST!), AND PROCEED THROUGH AN EXAMPLE OF A NEW PROGRAM.

(AS WITH MOST KRACKING ACTIVITIES, INITIALLIZED DISK FOR SAVING PIECES OF THE CODE AS THEY BECOME AVAILABLE).

ORDINARILY, WHEN YOU BOOT A 48K SLAVE DISK (A MASTER IS SLIGHTLY DIFFERENT, BUT WE'LL IGNORE THAT FOR THE TIME BEING), A THREE-STAGE PROCESS IS STARTED WHICH ENDS UP WITH THE DESIRED (HELLO) PROGRAM RUNNING. FIRST, THE CONTROLLER CARD ROM AT $C600-C6FF LOADS T0, S0 INTO PAGE 8, THEN JUMPS TO LOCATION $801. THIS IS A SHORT PROGRAM THAT LOADS ALL 10 SECTORS OF RWTS FROM T0, S0 THROUGH T0,S9 INTO PAGES $B6-BF ($B600-BFFF), THEN JUMPS TO LOCATION $B700. THIS PROGRAM, IN TURN, LOADS $1B (27) PAGES INTO $9D00-B5FF FROM T2, S4 THROUGH T0, SB (NOTE-THIS IS A "BACKWARDS LOAD" FOR SPEED. APPLE KNEW ABOUT IT, SO WHY DIDN'T DOS EVER USE IT FOR QUICKLOADING FILES???). AFTER A LITTLE HOUSEKEEPING, THE PROGRAM JUMPS TO THE DOS COLDSTART IN $9D84, WHICH RUNS OR EXECS THE HELLO PROGRAM. IN SUMMARY:

```
  CODE        # OF       DEST  NAME NEXT
LOCATION  SECT.  PAGE        JUMP
--------  -----  ----  -------      ----
C600-C6FF    1      08   STAGE 0     801
0801-08FF    9    B6-BF  STAGE 1    B700
B700-B7FF   27    9D-B5  STAGE 2    9D84
```

OF COURSE, IN A NONSTANDARD FORMAT INTENDED FOR PROTECTION, THINGS AREN'T NECESSARILY THE SAME. TO SEE THE DIFFERENCES, YOU NEED TO EXAMINE EACH STAGE SEPARATELY TO SEE WHAT IT DOES AND WHERE IT GOES.

THE THEORY OF BOOT-TRACING IS STRAIGHTFORWARD: FOLLOW THE BOOT PROCESS ONE STEP AT A TIME TO SEE WHERE IS LEADS YOU, BY CREATIVELY ALTERING THE THE CODE TO PREVENT IT FROM RUNNING AWAY FROM YOU.   IN SUMMARY, WE WILL:

1. READ IN THE STAGE 1 BOOT CODE, BUT NOT ALLOW IT TO EXECUTE,

2. ALTER THE FIRST STAGE BOOT SO IT WILL EXECUTE TO LOAD IN STAGE TWO, WHILE PREVENTING THE NEW STAGE FROM RUNNING,

3. IF NECESSARY, REPEAT THE PROCESS OF ALTERING, LOADING, AND HALTING UNTIL ALL THE STAGES OF THE BOOT HAVE BEEN EXAMINED AND UNDERSTOOD.

IN PRACTICE, THE FIRST TWO STEPS ARE RELATIVELY STANDARD, BUT STEP THREE CAN GET QUITE INVOLVED AS THE TRACE PROGRESSES.

THE TECHNIQUE FOR INTERRUPTING THE ORDERLY FLOW OF THE BOOT IS REFERRED TO AS "SETTING BREAK POINTS". THE TERMINOLOGY IS BORROWED FROM THE DARK AGES WHEN COMPUTERS HAD REAL FRONT PANELS WITH KNOBS AND SWITCHES AND LIGHTS, AND YOU COULD ACTUALLY "DIAL-IN" AN ADDRESS WHERE YOU WANTED THE COMPUTER TO HALT FOR EXAMINATION (IS ANYONE OUT THERE OLD ENOUGH TO SHARE MY FOND RECOLLECTION OF 'EXECUTE-STOP' AND 'FETCH-STOP' KNOBS?). SOPHISTICATED SYSTEMS WITH HIGH-LEVEL EXECUTIVE PROGRAMS STILL ALLOW THIS TODAY, BUT IN THE APPLE WE HAVE TO BE A LITLE MORE IMAGINATIVE.

IN ALL APPLE II SYSTEMS, THE INSTRUCTION SEQUENCE '4C 59 FF' OR JMP FF59 GOES TO THE RESET CODE AND PROVIDES A POSITIVE, PERMANENT STOPPING PLACE FROM ANYPLACE IN ASSEMBLY LANGUAGE CODE, AND HALTS WITH A WELL-DEFINED MACHINE STATE. WHENEVER WE WANT TO SET A "BREAKPOINT" IN THE APPLE, WE CAN REPLACE ANY THREE BYTES OF CODE WITH '4C 59 FF'.

TO BEGIN THE PROCESS, LETS LOOK AT SOME CODE FROM PART OF THE CONTROLLER CARD BOOT ROM:

```
  C600- A2 20         LDX #$20
C602- A0 00     LDY   #$00
C604- A2 03     LDX   #$03
         !
         !
C621- 20 58 FF    JSR   $FF58
C625- BD 00 01    LDA   $0100,X
C628- 0A        ASL
C629- 0A        ASL
C62A- 0A        ASL
C62B- 0A        ASL
C62C- 85 2B     STA   $2B
C62E- AA        TAX
C62F- BD 8E C0    LDA   $C08E,X
          !
          !
C658- A9 08     LDA   #$08
C65A- 85 27     STA   $27
C65C- 18        CLC
C65D- 08        PHP
C65E- BD 8C C0    LDA   $C08C,X
C661- 10 FB     BPL   $C65E
C663- 49 D5     EOR   #$D5
C665- D0 F7     BNE   $C65E
```

```
C667- BD 8C C0    LDA   $C08C,X
C66A- 10 FB     BPL   $C667
C66C- C9 AA     CMP   #$AA
C66E- D0 F3     BNE   $C663
C670- EA        NOP
C671- BD 8C C0    LDA   $C08C,X
C674- 10 FB     BPL   $C671
C676- C9 96     CMP   #$96
C678- F0 09     BEQ   $C683
           !
           !
C6E6- 91 26     STA   ($26),Y
C6E8- C8        INY
C6E9- D0 EE     BNE   $C6D9
C6EB- E6 27     INC   $27
C6ED- E6 3D     INC   $3D
C6EF- A5 3D     LDA   $3D
C6F1- CD 00 08    CMP   $0800
C6F4- A6 2B     LDX   $2B
C6F6- 90 DB     BCC   $C6D3
C6F8- 4C 01 08    JMP   $0801
C6FB- 00        BRK
C6FC- 00        BRK
C6FD- 00        BRK
```

  NOTICE THE INSTRUCTION 'JMP $0801' AT C6F8. THIS IS THE "LINK" TO STAGE 1 OF THE BOOT. IF WE COULD CHANGE IT TO 'JMP FF59', *EVERY* DISK WE BOOTED WOULD LOAD IN THE FIRST SECTOR, BEEP INTO THE MONITOR, AND OBLIGINGLY WAIT WHILE WE SNOOP THROUGH PAGE 8 TO OUR HEART'S CONTENT. SINCE THE PROGRAM IS IN ROM, WE CAN'T ALTER IT, BUT WE CAN COPY IT DOWN TO A COMPATIBLE LOCATION AND ALTER IT SO THAT THE PROGRAM HALTS INSTEAD OF CONTINUING WITH THE BOOT PROCESS. BECAUSE THE BOOT CODE HAS TO EXECUTE FROM AN} SLOT, IT CONTAINS A "WHERE ARE WE" ROUTINE AT C621-C62E TO FIND OUT WHAT ITS CURRENT LOCATION IS. HAPPILY FOR US, THIS KIND OF RELOCATABLE CODE WILL RUN MANY PLACES BESIDES THE C100-C7FF PERIPHERAL ROM SPACE (SEE THE REFERENCE MANUAL P. 81 FOR A DESCRIPTION OF THE "WHERE ARE WE" ROUTINE). MR.   XEROX'S FAMOUS MONITOR INSTRUCTIONS WHICH RELOCATE THE BOOT ROM CODE AND INSERT THE FIRST BREAKPOINT ARE:

```
      9600<C600.C6FFM
      96F8:4C 59 FF
```

  (NOTE-PAGE 96 IS NOT REQUIRED, BUT THE PAGE YOU USE MUST END IN 6 SO THAT SLOT 6 IS DECODED AS THE CONTROLLER CARD LOACTION). THE LAST FEW LINES OF THE (RELOCATED) BOOT ROM CODE NOW READ:

```
96F4- A6 2B     LDX $2B
96F6- 90 DB     BCC $96D3
96F8- 4C 59 FF    JMP $FF59
```

SO THAT TYPING:

```
      9600G
```

  WILL INITIATE A BOOT SEQUENCE FROM OUR CODE AT 9600 WHICH ENDS AT THE "BREAK POINT" AT $96F8, RATHER THAN CONTINUING THE BOOT. IF YOU TRY THIS, YOU'LL FIND THAT THE DISK IS STILL SPINNING, AND YOU CAN TURN IT OFF BY INCLUDING THE INSTRUCTION '2C E8 C0' (BIT C0E8) AT 96F8 BEFORE THE JMP FF59, OR YOU CAN JUST TYPE 'C0E8' FROM THE MONITOR. AFTER PAGE 8 HAS BEEN LOADED WITH THE STAGE 1

BOOT CODE, THE FUN BEGINS (UNTIL YOU GET GOOD AT THIS, IT'S A GOOD IDEA TO SAVE EACH PIECE OF BOOT CODE AS A BFILE ON A SPARE DISK BEFORE PROCEEDING. IT'S USUALLY EASIER THAN RUNNING THROUGH THE ENTIRE SEQUENCE EACH TIME A STEP DOESN'T WORK AS YOU EXPECT, AND IT WILL MAKE IT EASIER TO PRINT OUT A DISASSEMBLY OF THE CODE TO FIGURE OUT WHAT IT DOES.

AT THIS POINT, PAGE 8 MUST CONTAIN STAGE 1 OF THE BOOT WITH LOCATION $801 AS THE STARTING POINT. IF THE FIRST STAGE IS KOSHER, LOCATION $84A CONTAINS '6C FD 08', WHICH IS AN INDIRECT JUMP THROUGH THE LOCATION IN 8FD & 8FE. THIS IS THE EXIT POINT OF THE STAGE ONE BOOT, AND NORMALLY JUMPS TO B700 TO BEGIN READING IN THE CODE FOR STAGE 2 (THE B6 AT 8FE BECOMES B7 DURING THE 10-SECTOR LOAD). TO CONTINUE OUR MISSION, WE MUST LOCATE THE EXIT POINT OF THIS STAGE AND INSERT A BREAKPOINT.

```
0801- A5 27     LDA   $27
0803- C9 09     CMP   #$09
0805- D0 18     BNE   $081F
0807- A5 2B     LDA   $2B
0809- 4A        LSR
080A- 4A        LSR
080B- 4A        LSR
080C- 4A        LSR
080D- 09 C0     ORA   #$C0
080F- 85 3F     STA   $3F
0811- A9 5C     LDA   #$5C
0813- 85 3E     STA   $3E
0815- 18        CLC
0816- AD FE 08    LDA   $08FE
0819- 6D FF 08    ADC   $08FF
081C- 8D FE 08    STA   $08FE
081F- AE FF 08    LDX   $08FF
0822- 30 15     BMI   $0839
0824- BD 4D 08    LDA   $084D,X
0827- 85 3D     STA   $3D
0829- CE FF 08    DGC   $08FF
082C- AD FE 08    LDA   $08FE
082F- 85 27     STA   $27
0831- CE FE 08    DEC   $08FE
0834- A6 2B     LDX   $2B
0836- 6C 3E 00    JMP   ($003E)
0839- EE FE 08    INC   $08FE
083C- EE FE 08    INC   $08FE
083F- 20 89 FE    JSR   $FE89
0842- 20 93 FE    JSR   $FE93
0845- 20 2F FB    JSR   $FB2F
0848- A6 2B     LDX   $2B
084A- 6C FD 08    JMP   ($08FD)
084D- 00        BRK
084E- 0D 0B 09    ORA   $090B
0851- 07        ???
0852- 05 03     ORA   $03
0854- 01 0E     ORA   ($0E,X)
0856- 0C        ???
0857- 0A        ASL
0858- 08        PHP
0859- 06 04     ASL   $04
085B- 02        ???
085C- 0F        ???
```

```
085D- 00        BRK
       !
       !
08FD- 00        BRK
08FE- B6 09     LDX   $09,Y
```

  NONSTANDARD FORMATS CAN HAVE ANY NUMBER OF EXIT INSTRUCTIONS, AND THIS IS WHERE YOUR KNOWLEDGE OF ASSEMBLY LANGUAGE AND EXPERIENCE AT READING CODE WILL START TO PAY OFF. UNLESS THE FIRST STAGE IS RELATIVELY STANDARD, IT'S NECESSARY TO SPEND TIME EXAMINING AND TEARING APART THE CODE UNTIL YOU UNDERSTAND WHAT'S GOING ON. LOOK FIRST FOR A JUMP OR INDIRECT JUMP TO SOMEPLACE OUTSIDE OF PAGE 8, AND CHANGE THAT TO JMP FF59. IF NONE APPEARS, LOOK FOR A "JUMP THROUGH THE STACK" TRICK AS DESCRIBED IN THE ARCADG MACHINE FILE: FOR EXAMPLE, TO GO TO $BB00 THERE WILL BE, SOMEWHERE IN THE CODE, TWO "PHA'S" AND AN "RTS". THE FIRST PUSH ONTO THE STACK WOULD BE $BA; THE SECOND $FF. WHEN THE RTS IS EXECUTED, THE TWO BYTES ARE PULLED OFF THE STACK, INCREMENTED BY ONE TO BB00, AND JUMPED TO. IN ADDITION, MORE THAN ONE PAGE CAN BE LOADED UNDER STAGE 0, AND ACCESSED BY A RELATIVE BRANCH INSTRUCTIoN, SO YOU'LL HAVE TO EXAMINE <ALL> THE CODE LOADED IN (IT'S GOOD PRACTICE TO CLEAR OUT ALL OF MEMORY BEFORE STARTING; THIS WILL WORK IF DOS IS NOT ACTIVE:

```
800:0 N 801<800.BFFFM).
```

  WHEN YOU FIND THE EXIT POINT, MAKE IT A BREAKPOINT WITH '4C 59 FF' TO PREVENT THE CONTINUATION OF THE BOOT. BEFORE PROCEEDING, TAKE A GOOD LOOK AT ALL THE CODE TO BE SURE YOU UNDERSTAND WHERE THE NEXT STAGE LOADS, AND ANY UNUSUAL CONDITIONS OR INSTRUCTIONS.

  THE ALTERED PORTION OF CODE IS NOW:

```
0839- EE FE 08    INC   $08FE
083C- EE FE 08    INC   $08FE
083F- 20 89 FG    JSR   $FE89
0842- 20 93 FE    JSR   $FE93
0845- 20 2F FB    JSR   $FB2F
0848- A6 2B     LDX   $2B
084A- 4C 59 FF    JMP   $FF59
084D- 00        BRK
```

  THE THEORY NOW IS TO ALLOW THE BOOT TO PROCEED THROUGH ONE MORE STAGE, HALTING AFTER RWTS HAS BEEN READ IN, AND GIVING US A CHANCE TO EXAMINE THAT PORTION OF THE PROGRAM FOR ALTERATIONS. IF WE JUST REBOOTED WITH '9600G', THE ORIGINAL CODE WOULD OVERWRITE OUR ALTERED PAGE 8, SO WE HAVE TO ARRANGE IT SO THAT THE FIRST STAGE BOOT CODE IS SENT OFF INTO OBLIVION. REFERRING BACK TO THE BOOT CODE, LOCATION 9658 (ORIGINALLY C658) CONTAINS THE PAGE NUMBER WHERE T0, S0 LOADS IN, NORMALLY 08. CHANGING IT TO $20 WILL CAUSE T0, S0 TO LOAD INTO $2000 INSTEAD OF $0800, AND THE BOOT WILL CONTINUE THROUGH OUR ALTERED PAGE 8. NOTE THAT WE HAVE TO REMOVE THE FIRST BREAK POINT AT 96F8 AND RESTORE THE ORIGINAL JMP $0801:

```
         9658:20
         96F8:4C 01 08
```

  NOW, WHEN WE TYPE '9600G', THE BOOT CODE WILL LOAD T0, S0 INTO $2000-20FF, WHERE IT WON'T BOTHER US AT ALL, THEN JUMP TO 801 TO EXECUTE OUR CODE. AFTER RWTS HAS BEEN LOADED IN, INSTEAD OF JUMPING TO $B700 TO CONTINUE LOADING DOS, THE PROGRAM HITS THE (SECOND) BREAK POINT AT 84A AND HALTS.

THE FINAL PHASE OF THIS PROCESS IS TO LOCATE THE EXIT POINT FROM THIS AREA OF CODE, INSERT ANOTHER BREAKPOINT, AND EXAMINE ALL THE CODE LOADED IN BY STAGE 2. AGAIN, WE HAVE TO MAKE SURE THAT THE BOOT PROCESS DOESN'T OVERWRITE THE CHANGES, WHICH MEANS WE HAVE TO UNDERSTAND HOW THE DESTINATION ADDRESSES ARE SET UP IN STAGE 1. EVEN IN NORMAL DOS IT'S NOT OBVIOUS, BUT ENOUGH HEAD-SCRATCHING OR READING OF BENEATH APPLE DOS WILL REVEAL DHAT THE BYTE IN LOCATION 8FE IS ONE HIGHER THAN THE FIRST PAGE LOADED INTO, AND THE BYTE AT 8FF IS ONE LESS THAN THE NUMBER OF SECTORS TO BE LOADED. AS BEFORE, WE REMOVE THE PREVIOUS BREAKPOINT, ALTER THE DESTINATION OF THE REAL CODE LOADED IN UNDER THIS STAGE, AND SET THE NEW BREAKPOINT:

```
B700- 8E E9 B7    STX   $B7E9
B703- 8E F7 B7    STX   $B7F7
B706- A9 01      LDA   #$01
B708- 8D F8 B7    STA   $B7F8
B70B- 8D EA B7    STA   $B7EA
B70E- AD E0 B7    LDA   $B7E0
B711- 8D E1 B7    STA   $B7E1
B714- A9 02      LDA   #$02
B716- 8D EC B7    STA   $B7EC
B719- A9 04      LDA   #$04
B71B- 8D ED B7    STA   $B7ED
B71E- AC E7 B7    LDY   $B7E7
         !
         !
B738- 20 93 B7    JSR   $B793
B73B- A2 FF      LDX   #$FF
B73D- 9A         TXS
B73E- 8E EB B7    STX   $B7EB
B741- 4C C8 BF    JMP   $BFC8
B744- 20 89 FE    JSR   $FE89
B747- 4C 84 9D    JMP   $9D84
```

THE CHANGES ARE:

```
      84A:4C 00 B7
```

(WE CAN'T USE THE INDIRECT JUMP IN THE ORIGINAL, SMNCE WE HAVE REDIRECTED THE BOOT)

```
      8FE:20 09
```

(PAGE 20 OR ANYPLACE ELSE WHERE 10 PAGES OF CODE WON'T HURT ANYTHING)

```
      B747:4C 59 FF
```

(JMP 9D84 IS THE DOS COLD- START. THE JMP BFC8 IS A PATCH WHICH RETURNS WITH A JMP B744)

THE LAST FEW LINES OF CODE ARE NOW:

```
B741- 4C C8 BF    JMP $BFC8
B744- 20 89 FE    JSR $FE89
B747- 4C 59 FF    JMP $FF59
```

NOW TYPE '9600G', AND LET'S RECAP THE PROCESS THAT WILL OCCUR:

1. THE MODIFIED STAGE 0 CODE AT 9600-96FF WILL LOAD T0, S0 INTO PAGE 20 (SINCE WE DON'T WANT IT), THEN JUMP TO THE START OF OUR MODIFIED PAGE 8 AT 801.

2. THE MODIFIED PAGE 8 WILL LOAD T0, S0 THROUGH T0, S9 INTO PAGES 20 TO 2=,

THEN JUMP TO OUR MODIFIED CODE AT B700.

  3. THE MODIFIED CODE AT B700 WILL LOAD 27 SECTORS OF DOS INTO PAGES 9A-B5,
THEN HALT WHEN IT HITS THE BREAKPOINT AT B747.

```
---------------------------------------
***************************************
*                              *
*                              *
*                              *
*     THE BASICS OF KRACKING 109:     *
*                              *
*    BOOT-TRACING PART 2- RDF 1985    *
*                              *
*                              *
*                              *
***************************************
```

THIS IS THE SECOND PART OF THE BOOT-TRACING EPISODE--IT'S PROBABLY NOT HAZARDOUS
TO YOUR HEALTH TO READ THIS BEFORE YOU LOOK AT PART 1 (BASICS 108), BUT IT'LL
MAKE A MORE SENSE TO READ 108 FIRST IF YOU'RE NOT INTIMATELY FAMILIAR WITH THE
SUBJECT. THE THEORY (?) WAS ALL IN THE FIRST PART; THIS IS JUST AN EXAMPLE,
WITH A LOT OF DISASSEMBLED CODE, OF THE USE OF BOOT-TRACING TO LOOK AT (BUT NOT
REALLY TO KRACK) A NOT-TOO-UNUSUAL DISK: RDF 1985 FROM THOSE FUN-LOVING WAR
GAME FREAKS AT SSI. THIS LOADER/DOS APPEARS TO BE THE SUCCESSOR TO RDOS 2.1,
WHICH WAS THE LATE UNLAMENTED OPERATING SYSTEM THAT KEPT SO MANY DISKS FROM
BEING UNPROTECTED FOR SO LONG.

APPROACHING THIS DISK AS WE WOULD ANY OTHER, WE ENTER THE MONITOR AND SET THE
FIRST BREAKPOINT BY TYPING:

```
       9600<C600.C6FFM
       96F9:59 FF
       9600G
```

AFTER THE BEEP, AND C0E8 TO DESPIN THE DISK, 801LLLL GETS US THE FOLLOWING:

```
0801- A6 2B     LDX   $2B
0803- 8E 1F 02    STX   $021F
0806- A9 02     LDA   #$02
0808- 8D 20 02    STA   $0220
080B- 18        CLC
080C- 08        PHP
080D- BD 8C C0    LDA   $C08C,X
0810- 10 FB     BPL   $080D
0812- 49 D5     EOR   #$D5
0814- D0 F7     BNE   $080D
0816- BD 8C C0    LDA   $C08C,X
0819- 10 FB     BPL   $0816
081B- C9 AA     CMP   #$AA
081D- D0 F3     BNE   $0812
081F- EA        NOP
0820- BD 8C C0    LDA   $C08C,X
0823- 10 FB     BPL   $0820
0825- C9 B5     CMP   #$B5
0827- F0 09     BEQ   $0832
0829- 28        PLP
082A- 90 DF     BCC   $080B
```

```
082C- 49 AD     EOR   #$AD
082E- F0 20     BEQ   $0850
0830- D0 D9     BNE   $080B
0832- A0 03     LDY   #$03
0834- 84 2A     STY   $2A
0836- BD 8C C0    LDA   $C08C,X
0839- 10 FB     BPL   $0836
083B- 2A        ROL
083C- 85 3C     STA   $3C
083E- BD 8C C0    LDA   $C08C,X
0841- 10 FB     BPL   $083E
0843- 25 3C     AND   $3C
0845- 88        DEY
0846- D0 EE     BNE   $0836
0848- 28        PLP
0849- CD 20 02    CMP   $0220
084C- D0 BD     BNE   $080B
084E- B0 BC     BCS   $080C
0850- A0 00     LDY   #$00
0852- A9 00     LDA   #$00
0854- 85 47     STA   $47
0856- BD 8C C0    LDA   $C08C,X
0859- 10 FB     BPL   $0856
085B- 29 55     AND   #$55
085D- 0A        ASL
085E- 85 46     STA   $46
0860- BD 8C C0    LDA   $C08C,X
0863- 10 FB     BPL   $0860
0865- 29 55     AND   #$55
0867- 05 46     ORA   $46
0869- 45 47     EOR   $47
086B- 85 47     STA   $47
086D- 99 00 10    STA   $1000,Y
0870- C8        INY
0871- D0 E3     BNE   $0856
0873- BD 8C C0    LDA   $C08C,X
0876- 10 FB     BPL   $0873
0878- 29 55     AND   #$55
087A- 0A        ASL
087B- 85 46     STA   $46
087D- BD 8C C0    LDA   $C08C,X
0880- 10 FB     BPL   $087D
0882- 29 55     AND   #$55
0884- 05 46     ORA   $46
0886- 45 47     EOR   $47
0888- F0 02     BEQ   $088C
088A- D0 A4     BNE   $0830
088C- 4C 00 10    JMP   $1000
```

A FEW THINGS ARE WORTH POINTING OUT BEFORE WE CONTINUE THE TRACE. NOTICE THAT THE EARLY PART IS AN ADAPTATION OF THE BOOT ROM CODE: IF THE CARRY BIT IS CLEAR, IT'S LOOKING FOR D5 AA B5 TO READ IN THE ADDRESS FIELD (SOME THINGS NEVER CHANGE), IF THE CARRY IS SET, D5 AA AD IS BEING SOUGHT FOR THE DATA FIELD PROLOG. AFTER VERIFYING THE VOLUME, TRACK, AND SECTOR (832-846), WE READ IN A SINGLE "PSEUDO-SECTOR" IN 4+4 NIBBLIZING, STORING IT AT $1000. IF THE CHECKSUM IS RIGHT ($888), THEN WE JUMP TO 1000 TO CONTINUE THE BOOT. IF YOU'VE BEEN KEEPING UP, YOU KNOW THE NEXT SERIES OF MONITOR INSTRUCTIONS TO SET BREAKPOINT #2:

```
         96F9:01 08
         9659:20
         088C:4C 59 FF
         9600G

THE CODE LOADED INTO PAGE $10 IS:

1000- D8        CLD
1001- D8        CLD
1002- A9 00     LDA   #$00
1004- 8D F2 03    STA   $03F2
1007- A9 E0     LDA   #$E0
1009- 8D F3 03    STA   $03F3
100C- 49 A5     EOR   #$A5
100E- 8D F4 03    STA   $03F4
1011- A9 4C     LDA   #$4C
1013- 8D D0 03    STA   $03D0
1016- A9 00     LDA   #$00
1018- 8D D1 03    STA   $03D1
101B- A9 BD     LDA   #$BD
101D- 8D D2 03    STA   $03D2
1020- AD 1F 02    LDA   $021F
1023- 8D D3 03    STA   $03D3
1026- A9 01     LDA   #$01
1028- 8D D4 03    STA   $03D4
102B- A9 03     LDA   #$03
102D- 8D 20 02    STA   $0220

1030- A9 BD     LDA   #$BD
1032- 8D 99 10    STA   $1099

1035- 18        CLC
1036- 08        PHP
1037- BD 8C C0    LDA   $C08C,X
103A- 10 FB     BPL   $1037
103C- 49 D5     EOR   #$D5
103E- D0 F7     BNE   $1037
1040- BD 8C C0    LDA   $C08C,X
1043- 10 FB     BPL   $1040
1045- C9 AA     CMP   #$AA
1047- D0 F3     BNE   $103C
1049- EA        NOP
104A- BD 8C C0    LDA   $C08C,X
104D- 10 FB     BPL   $104A
104F- C9 B5     CMP   #$B5
1051- F0 09     BEQ   $105C
1053- 28        PLP
1054- 90 DF     BCC   $1035
1056- 49 AD     EOR   #$AD
1058- F0 20     BEQ   $107A
105A- D0 D9     BNE   $1035
105C- A0 03     LDY   #$03
105E- 84 2A     STY   $2A
1060- BD 8C C0    LDA   $C08C,X
1063- 10 FB     BPL   $1060
1065- 2A        ROL
1066- 85 3C     STA   $3C
```

```
1068- BD 8C C0    LDA   $C08C,X
106B- 10 FB     BPL   $1068
106D- 25 3C     AND   $3C
106F- 88        DEY
1070- D0 EE     BNE   $1060
1072- 28        PLP
1073- CD 20 02    CMP   $0220
1076- D0 BD     BNE   $1035
1078- B0 BC     BCS   $1036
107A- A0 00     LDY   #$00
107C- A9 00     LDA   #$00
107E- 85 47     STA   $47
1080- BD 8C C0    LDA   $C08C,X
1083- 10 FB     BPL   $1080
1085- 29 55     AND   #$55
1087- 0A        ASL
1088- 85 46     STA   $46
108A- BD 8C C0    LDA   $C08C,X
108D- 10 FB     BPL   $108A
108F- 29 55     AND   #$55
1091- 05 46     ORA   $46
1093- 45 47     EOR   $47
1095- 85 47     STA   $47

1097- 99 00 10    STA   $1000,Y

109A- C8        INY
109B- D0 E3     BNE   $1080
109D- BD 8C C0    LDA   $C08C,X
10A0- 10 FB     BPL   $109D
10A2- 29 55     AND   #$55
10A4- 0A        ASL
10A5- 85 46     STA   $46
10A7- BD 8C C0    LDA   $C08C,X
10AA- 10 FB     BPL   $10A7
10AC- 29 55     AND   #$55
10AE- 05 46     ORA   $46
10B0- 45 47     EOR   $47
10B2- F0 02     BEQ   $10B6
10B4- D0 A4     BNE   $105A
10B6- EE 99 10    INC   $1099
10B9- AD 99 10    LDA   $1099
10BC- C9 C0     CMP   #$C0
10BE- F0 06     BEQ   $10C6
10C0- EE 20 02    INC   $0220
10C3- 4C 35 10    JMP   $1035
10C6- A9 BA     LDA   #$BA
10C8- 85 00     STA   $00
10CA- A9 BC     LDA   #$BC
10CC- 85 01     STA   $01
10CE- A9 01     LDA   #$01
10D0- 85 03     STA   $03
10D2- A9 00     LDA   #$00
10D4- 85 04     STA   $04
10D6- A9 06     LDA   #$06
10D8- 85 05     STA   $05
10DA- 20 D0 03    JSR   $03D0
10DD- A9 F0     LDA   #$F0
```

```
10DF- 85 36     STA   $36
10E1- A9 FD     LDA   #$FD
10E3- 85 37     STA   $37
10E5- 4C 00 BA    JMP   $BA00
```

  THE EARLY PART FROM 1002-102D SETS UP THE 3D0-3FF REGION AS VECTORS FOR THE "DOS" CALLS TO BE MADE, THEN STORES $BD IN $1099 FOR THE PAGE NUMBER TO BEGIN LOADING IN THE NEXT PORTION OF THE BOOT. AFTER THAT, THE CODE FROM 80C-847 IS MIRRORED TO LOAD IN THE NEXT STAGE. THE THREE LINES AT 10B6-10BF INDICATE THAT THE LOAD CONTINUES UNTIL PAGES BD, BE, AND BF HAVE BEEN LOADED, THEN QUITS AT PAGE $C0. YOU WOULD NORMALLY EXPECT TO FIND A "JMP BD00" AS THE EXIT POINT FROM THIS STAGE OF THE BOOT; INSTEAD THERE IS A "JMP BA00" AT 10E5. THE REASON IS THAT LINES 10C6- 10DA CALL THE NEWLY-LOADED LOADER ROUTINE AT BD00 THROUGH THE VECTOR AT 3D0.   BY LOOKING AT THE SETUP FOR THAT LOAD, WE CAN LEARN A LITTLE ABOUT THE LOADER. THE IMPORTANT PARTS OF ANY LOADER ROUTINE ARE THE DESTINATION PAGE, THE LENGTH OF THE LOAD, AND THE TRACK AND SECTOR TO BEGIN LOADING FROM. IN THIS CASE, THE FIRST AND LAST DESTINATION PAGE ARE LOADED INTO LOCATIONS 0 AND 1, AND THE TRACK AND SECTOR IN 4 AND 5. AFTER THAT, A CALL TO THE 3D0 VECTOR JUMPS MERRILY UP TO BD00, WHICH IS THE "RWTS" ROUTINE FOR THIS PROGRAM.

  WE CAN VIEW ALL OF THAT BY SETTING THE NEXT BREAKPOINT AT 10E5 AND REBOOTING FOR WHAT IS HOPEFULLY THE LAST TIME:

```
         086F:20
         088C:4C 00 10
         10E5:4C 59 FF
         9600G
```

  SINCE THE EXIT POINT SAID JUMP BA00, LETS LOOK AT THAT CODE:

```
BA00- D8        CLD
BA01- 4C 00 BC    JMP   $BC00
BA04- A5 8D     LDA   $8D
BA06- 9E        ???
```

  THE REST OF THE PAGE IS OF NO INTEREST, SO LET'S FOLLOW THE JUMP TO BC00:

```
BC00- AD 00 08    LDA   $0800
BC03- C9 EA     CMP   #$EA
BC05- D0 0D     BNE   $BC14
BC07- A2 05     LDX   #$05
BC09- BD BE BC    LDA   $BCBE,X
BC0C- 95 00     STA   $00,X
BC0E- CA        DEX
BC0F- 10 F8     BPL   $BC09
BC11- 20 D0 03    JSR   $03D0
BC14- AD D7 BC    LDA   $BCD7
BC17- D0 48     BNE   $BC61
BC19- AD 81 C0    LDA   $C081
BC1C- A9 00     LDA   #$00
BC1E- 8D F2 03    STA   $03F2
BC21- A9 BC     LDA   #$BC
BC23- 8D F3 03    STA   $03F3
BC26- A9 19     LDA   #$19
BC28- 8D F4 03    STA   $03F4
BC2B- A9 00     LDA   #$00
BC2D- 8D 11 03    STA   $0311
BC30- A9 00     LDA   #$00
```

```
BC32- 8D 13 03    STA   $0313
BC35- A2 05     LDX   #$05
BC37- BD AC BC    LDA   $BCAC,X
BC3A- 95 00     STA   $00,X
BC3C- CA        DEX
BC3D- 10 F8     BPL   $BC37
BC3F- 20 D0 03    JSR   $03D0
BC42- A9 00     LDA   #$00
BC44- 8D 12 03    STA   $0312
BC47- 20 00 A8    JSR   $A800
BC4A- EE D7 BC    INC   $BCD7
BC4D- AD 12 03    LDA   $0312
BC50- C9 02     CMP   #$02
BC52- F0 70     BEQ   $BCC4
BC54- A2 05     LDX   #$05
BC56- BD B2 BC    LDA   $BCB2,X
BC59- 95 00     STA   $00,X
BC5B- CA        DEX
BC5C- 10 F8     BPL   $BC56
BC5E- 20 D0 03    JSR   $03D0
BC61- AD 12 03    LDA   $0312
BC64- D0 5E     BNE   $5CC4
BC66- A9 25     LDA   #$25
BC68- 85 03     STA   $03
BC6A- A9 00     LDA   #$00
BC6C- 8D 10 03    STA   $0310
BC6F- 2C 10 C0    BIT   $C010
BC72- A2 06     LDX   #$06
BC74- A9 80     LDA   #$80
BC76- 95 F6     STA   $F6,X
BC78- CA        DEX
BC79- 10 FB     BPL   $BC76
BC7B- AD 50 C0    LDA   $C050
BC7E- AD 54 C0    LDA   $C054
BC81- AD 57 C0    LDA   $C057
BC84- AD 09 03    LDA   $0309
BC87- F0 06     BEQ   $BC8F
BC89- AD 52 C0    LDA   $C052
BC8C- 4C 92 BC    JMP   $BC92
BC8F- AD 53 C0    LDA   $C053
BC92- A9 00     LDA   #$00
BC94- 8D 14 03    STA   $0314
BC97- 8D 07 03    STA   $0307
BC9A- 20 00 65    JSR   $6500
BC9D- AD 11 03    LDA   $0311
BCA0- D0 8E     BNE   $BC30
BCA2- AD 12 03    LDA   $0312
BCA5- C9 01     CMP   #$01
BCA7- F0 1B     BEQ   $5CC4
BCA9- 4C 00 E0    JMP   $E000

BCAC- A8        TAY
BCAD- BB        ???
BCAE- 00        BRK
BCAF- 01 01     ORA   ($01,X)
BCB1- 00        BRK

BCB2- A8        TAY
```

```
BCB3- BB        ???
BCB4- 00        BRK
BCB5- 01 13     ORA   ($13,X)
BCB7- 00        BRK

BCB8- A8        TAY
BCB9- BB        ???
BCBA- 00        BRK
BCBB- 01 15     ORA   ($15,X)
BCBD- 00        BRK

BCBE- 08        PHP
BCBF- 14        ???
BCC0- 00        BRK
BCC1- 01 03     ORA   ($03,X)
BCC3- 00        BRK

BCC4- A2 05     LDX   #$05
BCC6- BD B8 BC    LDA   $BCB8,X
BCCB- CA        DEX
BCCC- 10 F8     BPL   $BCC6
BCD1- 20 00 A8    JSR   $A800
BCD4- 4C 54 BC    JMP   $BC54
```

FINALLY, HERE'S THE MEAT OF THE PROGRAM. BC00 IS A TEST TO SEE IF IT'S THE FIRST TIME THROUGH--LOCATION 800 IS 01 THE FIRST TIME, SO WE TRANSFER THE 5 VALUES FOUND AT BCBE-BCC3 INTO LOCATIONS 0-5, THEN CALL THE LOADER ROUTINE, AND PAGES 8-14 ARE LOADED FROM TRACK 3, SECTOR 0. AT BC1C-BC2B, WE SET THE RESET VECTOR TO RETURN TO BC00 (AND RESTART THE GAME) WHENEVER RESET IS PRESSED (THE REFERENCE MANUAL TELLS YOU HOW ON P. 37; IT'S P. 82 IN THE IIE MANUAL, IF YOU GOT RIPPED OFF FOR THAT ONE). NEXT, PAGES A8-BB ARE LOADED FROM T1, S0, AND THE GAME BEGINS IN EARNEST.

ORDINARILY, THIS IS ABOUT AS FAR AS BOOT-TRACING CAN TAKE YOU INTO THE ORGANIZATION OF A DISK. JUST FOR THE EXERCISE, HOWEVER, LET'S PRETEND WE REALLY WANT TO FIND OUT WHAT GETS LOADED INTO PAGES A8-BB BEFORE THE GAME STARTS. THE BREAKPOINT GOES IN...

```
          10E5:4C 00 BA
          1099:20
          10BD:23 (UNLESS YOU KNOW
                 IT'S SAFE, KEEP
                 THE # OF PAGES THE
                 SAME)
          BC47:4C 59 FF
```

SEE WHAT THAT CODE LOOKS LIKE BEFORE IT HAS A CHANCE TO UNSCRUNCH A PICTURE OR WHATEVER ELSE IT IS GOING TO DO. ALSO, NOTICE THAT THE CODE AT BC54 AND AT THE ALTERNATIVE DESTINATION OF BCC4 BOTH LOAD OVER PAGES A8-BB. IF NECESSARY, WE COULD CONTINUE WITH THIS PROCESS, PUTTING A BREAKPOINT AFTER EACH LOAD, UNTIL WE HAVE EXAMINED, SAVED, OR ALTERED EVERY ACCESSIBLE PART OF THE PROGRAM.

SO MUCH FOR THE "EASY" PART--NOW THE HARD WORK BEGINS. WE HAVE LEARNED ABOUT ALL WE NEED TO KNOW ABOUT THE LOADER, BUT NOW WE HAVE TO FIND A WAY TO PUT ALL OF THIS INTO A FORMAT WHICH UNLESS WE CAN USE DOS ON THE LANGUAGE CARD, CONSIDERING THAT ALL OF THESE FILES LOAD RIGHT OVER THE MIDDLE OF DOS), OR CONVERT THE 4+4 NIBBLIZED SECTORS INTO STANDARD DOS 3.3 SECTORS (ALSO NOT TOO EASY, CONSIDERING THAT WE ONLY HAVE 3 PAGES FOR THE "DOS").

  IN CONCLUSION, YOU SHOULD BE AWARE THAT THE TECHNIQUES DESCRIBED HERE WORK EQUALLY WELL ON AN APPLE IIE (I HELD OUT FOR ALMOST 8 MONTHS, BUT NOW I CAN CONTEND THAT ALL THE TYPOS IN THIS EPISODE ARE THE RESULT OF HAVING FUNNY KEYS LIKE "][" ON AN APPLE KEYBOARD AFTER 5+ YEARS OF TYPING ON GOOD OLD APPLE II S/N 3603).   STAY TUNED FOR AN EXAMPLE WHICH IS A LITTLE MORE COMPLICATED THAN THIS ONE, AND REQUIRES CHANGES IN THE BOOT-TRACE TECHNIQUE.   ALSO IN THE WINGS IS (WHAT ELSE) HARDWARE MODIFICATIONS TO THE IIE (NO MOTHER BOARD SURGERY, I PROMISE) TO ALLOW KRAKROMS, HARD RESETS, AND KREATIVE KRACKING USE OF THE 64K 80-COLUMN BOARD.

```
==============================================================================
DOCUMENT mac2info.app
==============================================================================

<%=-----------------------------------------------------------------------=%>
<%=--=%>                 The Macintosh II                      <%=--=%>
<%=--=%>                                                 <%=--=%>
<%=--=%>                   Presented by                        <%=--=%>
<%=--=%>          The Dragons Den BBS/Cat-Fur/AE                     <%=--=%>
<%=--=%>                  (617) 922-1917                       <%=--=%>
<%=--=%>                   March 2, 1987                       <%=--=%>
<%=--=%>                                                 <%=--=%>
<%=--=%>                    Written by                         <%=--=%>
<%=--=%>                 The Dragonslayer                      <%=--=%>
<%=--=%>                                                 <%=--=%>
<%=-----------------------------------------------------------------------=%>
```

Powerful Open Macintosh Expands Applications

AppleWorld, Los Angeles, California, March 2, 1987. Apple Computer, Inc. today introduced a high-performence, open architecture member of the Macintosh personal computer family, the Macintosh II. The new Macintosh offers users high speed, expansion and fleibility. It modular design and open architecture permit a number of display options, including color displays, and th ability to incorporate add-in cards from Apple and third party for additional functionality.

This top-of-the-line model is intended for advanced applications in business, desktop publishing, higher education and engineering enviroments.

"Because of its power and expandability, The Macintosh II strengthens Apple's position in markets in which we are already participating and extends the Macintosh personal computer family into new markets," said William V. Campbell, executive vice president U.S. Sales and Marketing.

At introduction, the Macintosh II operates most existing Macintosh applications up to 4 times faster than the Macintosh Plus. The Macintosh II offers upward compatability with the majority of existing applications. Apple is working closely with third-party hardware and software developers to ensure that a wide range of software, peripherals and add-on cards are developed to take full advantage of the advanced features of te Macintosh II.

Macintosh II Specifications

The Macintosh II is based on the 32-bit Motorola 68020 microprecessor operating at 16 megaherta (MHz). It includes a floating point arithmetic chip, the 68881, that can perform mathematical operations up to 200 times faster then the 68020. These features let the Macintosh II process at a speed of 2 million instructions per second (2 MIPS). The Macintosh II also features transfer rates greater then 1 megabyte(MB) per second over its Small Computer Systems Interface(SCSI) interface.

The Macintosh II come standard with 1 MB of random-access memory(RAM), expandable to 8 MB on the logic board.   Additional RAM expansion of up to 1.5 gigabytes(GB) can be achived with add-in boards.

The Macintosh II provides Macintosh Plus-compatable ports for a SCSI connection, two RS-422 serial ports, an external SCSI disk drive interface and a sound port with four-voice stereo capability. Like all Macintoh computers, the Macintosh II has the AppleTalk network built in.

In addition, the Macintosh II includes six slots that use the high-performance NuBus protocols. NuBus is a processor-independent, industry standard bus that supports 8-, 16- 32-bit data paths. It permits the fast transferof large quantities of data between add-on cards and he logic board. NuBus features fair arbitration and geographical addressing. Te two characteristics let the add-on cards "identify" themselves so, unlike other computer systems thereis no need to set dip switches to configurethe ystem. Because NuBus lets add-in cards be placed in any slot, thre is exceptional flexibility and ease associated with system configuration. Th six slots let the Macintosh II operate a wide range of performance-driven, deanding applicatons and expand as users' needs expand.

The video interface is provided by the Macintosh II video card which fits in one of the slots. The card can drive either of the high-resolution monitors introduced today. In its standard configuration, the card can simultaneously generate 16 colors or shades of gray from a standard palette of more than 16 million colors. With the addition of the Macintosh II video Card Expansion Kit, the card can generate up to 256 colors or shades of gray from te same palette.

Users may choose a 12-inch, high-resolution, monochrome monitor or a 13-inch, high-resolution red-green-blue(RGB) color monitor. Both display units feature 640 x 480 pixel resolution and utilize an analog input format.   This formatlets the monochrome monitor display millions of gray values and he color monitor display millions of colors or gray values.

The monochrome monitor, which is capable of displaying the full width and over half the length of a page, suits a need in productivity applications such as word processing, spreadsheets and business graphics.

The RGB monitor combines the full-width viewing area with te unique capability of displaying high-resolution text and graphics in both colorand black-and-white. This provides the Macintosh II user with a versatile, high-performance monitor capable of satisfying a broad spectrum of user needs from word processing to advanced graphics. A tilt-and-swivel monitor stand is available as an option for the high-resolution monitors. Users can configure the Macintosh II with multiple monitors by adding video cards in slots. Various monitors and video cards are also available from third parties. The Macintosh II also includes the Apple Desktop Bus(ADB) standard interface for input peripherals. ADB is also used on the Macintosh SE as well as the Apple //gs. The ADB lets users connect up to 16 input devices concurrently, including such peripherals as a keyboard, mouse or graphics tablet. Users may also choose from two Apple keyboards: the Apple Keyboard includes a typewriter style layout, a numeric keypad and cursor keys: and the Apple Extended Keyboard includes the numeric keypad, function keys and special purpose keys for sing alternative operating systems, such as MS-DOS or terminal emulation programs. Keyboards are packaged and sold separately.   The Macintosh II can internally accomodate, simultaneously, up to two 800 kilobyte(KB) floppy disk drives and one 20, 40, 80 MB hard disk. Both the 40 and 80 MB hard disks feature a very fast access time of less then 30 milliseconds(ms). In addition, up to six storage devices can be daisy-chained through the external SCSI port. For those users who want to back up critical data from thier hard disks, Apple also introduced an optional SCSI 40 MB tape backup unit, which provides file and image backup on preformatted, one-quarter-inch tpae cartridges. Apple also introduced the Apple EtherTalk interface Card, which provides direct connectivity to Ethernet networks for the Macintosh II. Apple will support AppleTalk network architecture and A/UX (Apple's UNIX product) networking software enviroment for use with the EtherTalk Card. Third party vendors are expected to provide software support allowing connectivity to other enviroments. The EtherTalk product will be available in the second half of 1987.

Alternative Operating Enviroments

A/UX, a version of AT&T UNIX

Apple also announced today that it will offer a version of the UNIX operating system for the Macintosh II. This operating system is widely used in universities, in government and by technical professionals. An optional Motorola 68851 paged memory management unit (PMMU) is required for A/UX and will be available from Apple. Unisoft Systems developed a significant portion of A/UX under contract with Apple. A/UX is a full implementation of the AT&T UNIX, System V, Release 2 Version 2 operating system and includes features from Berkeley's 4.2 BSD version. The featurs incorporated from 4.2 BSD provide easy portability of programs from 4.2 BSD to A/UX and andvanced communications capabilities.

A Macintosh II running A/UX offers the tradidional user interface of a UNIX operating system: a high-powered command line interpreter. Standard UNIX System V applications can be easily ported to A/UX. Additionally, a key enhancement from Apple lets A/UX developers have full access to the Macintosh Toolbox. A/UX applications can therefore have the complete look and feel of Macintosh programs. New applications, properly designed, can operate in both enviroments.

A/UX also offers, through add-in cards, connections to Ethernet, AppleTalk and serial communications networks using standard UNIX communications and electronic mail systems. It can also act as a server or a client on a Sun Microsystems Network File Systems (NFS) Ethernet network. The Apple EtherTalk Interface Card provides direct connectivity to Ethernet networks for the Macintosh II. A/UX is expected to ship this sumer. Pricing and licensing will be announced in May.

MS-DOS

Apple's goal is to provide data file inter-change with other operating systems, to provide MS-DOS data file compatability, Apple is introducing InterFile, file transfer software, a 5.25-inch MS-DOS floppy disk drive and drive controler cards. In addition, MS-DOS coprocessor cards for te Macintosh II and the Macintosh SE are available from third parties.

For example, users who purchase the 5.25-inch drive and controler card from Apple can read in a Lotus 1-2-3 data file so it can be used in a spreadsheet program, such as Microsoft Excel, on the Macintosh. Or, users who choose a coprocessor card from a third party can run dBase III or Lotus 1-2-3 in a window on the Macintosh screen.

International Models

Apple is Simultaneously introducing the Macintosh II available in 15 localized versions in 10 different languages, including English, French, German, Spanish, Flemish, Norwegian, Japanese, Dutch, Swedish and Italian. The Macintosh II features a universal power supply that permits operation with all common voltage.

Price and Availablity

The Macintosh II will be available in May in two configurations in a new platinum color: a basic system, including 1 MB of RAM and one 800KB floppy disk drive is offeredat a suggested retail price of $3,898, inclusing keyboard; a second configuration, including 1 MB of RAM, one 800KB floppy disk drive and one 40 MB internal SCSI hard disk is listed at a suggested retail price of $5,498, inclusing keyboard. Many of the other products introduced today are available as options for the Macintosh II.

Macintosh Technology

Macintosh personal computer technology -- manfested by ease of use, graphics and unique functionality -- features a very high level of software consistancy and tight intergration across all applications, resulting in low requirements for user support and training.

These attributes have contributed to the widespread acceptance of the Macintosh personal computer family accross all sizes of business and in higher education and has increased momentum by third-party developers over the past year.

Over one million Macintosh computers handle business, education and consumer applications.

Call These fine boards.

```
Dragons Den...................(617) 922-1917
Capital Connection............(916) 448-3402
Capital Connection ][.........(716) 473-8051
```

==
DOCUMENT maccrack.app
==

THE BYTE'S MAC-CRACK #1

CHAPTER 1 -- PROTECTION METHODS
===============================

IN THESE EARLY DAYS OF MAC-CRACKING, THERE ARE ONLY A FEW MAIN METHODS OF PROTECTING PROGRAMS. ONE OF THE MOST POPULAR METHODS OF PROTECTING A DISK INCLUDES HIDING AN INVISIBLE FILE ON THE DISK. THE APPLICATION THEN CHECKS TO MAKE SURE THAT THE FILE IS ON THE DISK, AND IF IT DOESN'T FIND THE FILE IT WILL CRASH, HANG, OR TELL YOU TO "INSERT MASTER". [MULTIPLAN USES THIS METHOD]. AN INVISIBLE FILE CANNOT BE COPIED WITH THE FINDER, BUT IT WILL BE COPIED WHEN A FULL DISK COPIER IS USED (IE. DISKCOPY, DISKUTIL, OR BLOCKSMITH). TO STOP PEOPLE FROM COPYING THE WHOLE DISK, THE DISKPROTECT BYTE ON BLOCK #2 IS SET TO $40. DISKCOPY (THE COPY PROG WHICH EVERYONE GETS ON THEIR SYSTEM DISK) CHECKS THAT BYTE, AND IF IT IS A $40, DISKCOPY WILL SPIT OUT THE DISK AND SAY "THAT DISK IS COPY-PROTECTED!" (SHAME ON YOU).

NOT ONLY DOES EACH DISK HAVE A PROTECTION BYTE, BUT EACH FILE IN THE DIRECTORY ALSO HAS ONE. THIS BYTE (CALLED THE "ATTRIBUTE" BYTE BY EXAMINEFILE) IS TRICKY. ONCE IT HAS BEEN SET (BY EXAMINEFILE FOR EXAMPLE), THE ONLY WAY TO RESET IT (RIGHT NOW) IS TO USE A BLOCK EDITOR (SOMETIMES CALLED A 'DISK ZAP'). YOU CAN USE EXAMINEFILE TO SEE IF A FILE IS PROTECTED THIS WAY (THE ATTRIBUTE BYTE WILL BE A $40) OR YOU CAN JUST TRY TO MOVE, TRASH, OR DUPLICATE THE SUSPECT FILE. IF YOU GET THE MESSAGE: "THAT MAY NOT BE DUPLICATED OR MOVED" THEN YOU KNOW IT'S PROTECTED BY THAT BYTE.

ALL OF THE ABOVE METHODS OF PROTECTION ARE NO CONTEST FOR DISKUTIL AND TWO DRIVES, BUT IT'S NICE TO BE ABLE TO COPY WITH ONE DRIVE AND TO PUT STUFF ON THE SAME DISK....

UNFORTUNATELY, A FEW PROGRAMS [MILLIONAIRE, THINKTANK] ARE NOW USING SOME STRANGE TRACKS THAT WILL NOT COPY WITH DISKUTIL. I HAVE BEEN TOLD THAT THE BLOCKSMITH TYPE OF COPIER WILL CHURN THROUGH THAT TYPE, BUT I HAVEN'T BEEN ABLE TO GET MY HANDS ON AN ORIGINAL TO TRY IT.

THE BYTE

=======================================

THE BYTE'S MAC-CRACK #2

=======================================

CHAPTER 2 -- CRACKING METHODS
=============================

REQUIRED TOOLS:
- -BLOCK EDITOR V1.01
- -SETFILE
- -EXAMINEFILE
- -DISKUTIL
- -BLOCKSMITH OR EQUIVALENT
- -BLANK DISKS

FIRST, I'LL EXPLAIN WHAT WE ARE TRYING TO DO, THEN I'LL GIVE SPECIFICS. OUR

MAIN OBJECTIVE IS TO MAKE A PROTECTED DISK COPYABLE WITH THE FINDER, BUT MAKING A DISK COPYABLE WITH DISKCOPY WILL ALSO BE SUFFICIENT (THE EQUIVALENT OF THE "COPYA" TO THOSE OF YOU IN APPLE //-LAND). IF DISKCOPY SAYS THAT A DISK IS COPY-PROTECTED, WE MUST DISABLE THE DISKPROTECT BYTE ON BLOCK #2. IF THE FINDER SAYS THAT ANY FILES ON THE DISK CANNOT BE MOVED/DUPLICATED, WE MUST RESET THE ATTRIBUTE BYTE IN THE DIRECTORY FOR EACH FILE. IF DISKUTIL OR BLOCKSMITH CANNOT COPY THE ORIGINAL, THEN YOU WILL HAVE TO MAKE ANY INVISIBLE FILES VISIBLE (WITH SETFILE) AND MOVE ALL THE FILES TO ANOTHER DISK. HOPEFULLY THE FILES WILL NOT BE LOCATION-DEPENDANT.

OK, LET'S CRACK THE BASIC GENERIC PROTECTION OF MOST PROGRAMS. FIRST COPY THE ORIGINAL WITH DISKUTIL OR WHATEVER, THEN USE SETFILE TO MAKE ALL INVISIBLE FILES VISIBLE (THIS IS OPTIONAL IF YOU WILL END UP WITH A "DISKCOPY", OR "COPYA", CRACK). NOW USE THE BLOCK EDITOR TO READ IN BLOCK #2, AND LOOK AT BYTE $0A (10 DECIMAL). THIS BYTE WILL MOST PROBABLY BE A $40, BUT WHATEVER IT IS, CHANGE IT TO A $00. NOW THE DISK CAN BE COPIED WITH DISKCOPY. NEXT READ IN BLOCK #4, WHICH SHOULD BE THE FIRST BLOCK OF THE DIRECTORY. A SHORT DESCRIPTION OF THE FORMAT OF EACH FILE ENTRY IS IN ORDER. (NOTE THAT THE LENGTH OF EACH ENTRY DEPENDS ON THE LENGTH OF THE FILE NAME, WHICH IS LAST). THE ZEROETH BYTE OF EACH ENTRY IS THE ATTRIBUTE BYTE WITH THE HIGH BIT SET. THE NEXT BYTE IS THE VERSON # (I THINK) AND IS USUALLY ZERO. THE NEXT FOUR WORDS (1 WORD = 2 BYTES) ARE THE FILE TYPE & CREATOR; THEY CAN BE CHANGED WITH SETFILE, BUT IT SHOULDN'T BE NECESSARY.

NEXT COMES 20 WORDS (40 BYTES) OF DIFFERENT INFO ON THE FILE, NONE OF IT VERY IMPORTANT. THE NEXT BYTE IS THE LENGTH OF THE FILENAME AND THEN THE FILENAME IN POSITIVE ASCII. IMMEDIATELY FOLLOWING THE NAME (OR ON THE NEXT EVEN BYTE) IS THE ZEROETH BYTE OF THE NEXT ENTRY...

SINCE THE ATTRIBUTE BYTE (AS DISPLAYED BY EXAMINEFILE) IS NORMALLY A ZERO, THE NORMAL VALUE OF THIS ZEROETH BYTE IS $80 (ZERO WITH THE HIGH BIT SET).

FOR A PROTECTED FILE, THIS BYTE WILL BE A $C0. CHANGE IT TO AN $80. THIS FILE CAN NOW BE MOVED/DUPLICATED. THE TRICK TO THIS IS FINDING THE ATTRIBUTE BYTE FOR THE PROTECTED FILES. THE EASIEST WAY IS TO LOOK FOR FILENAMES IN THE ASCII DUMP ON THE RIGHT OF THE SCREEN (YOU ARE USING BLOCK EDITOR AREN'T YOU), AND THEN LOOK AT THE BYTE IMMEDIATELY AFTER THE NAME. IF IT'S A $C0, THEN THE ODDS ARE GOOD THAT IT SHOULD BE AN $80.

YOU WILL GET USED TO FINDING WHAT YOU ARE LOOKING FOR....

ONCE YOU HAVE DE-PROTECTED THE DISK AND ITS FILES, THE DISK SHOULD BE COPYABLE WITH DISKCOPY. HOPEFULLY, IT WILL ALSO WORK IF YOU COPY THE FILES ALONE (ALL FILES MUST BE VISIBLE), BUT NOT NECESSARILY. TAKE PFS FOR EXAMPLE: THERE ARE TWO PROTECTED FILES ("PFS FIL E" & "PFS REPORT") AND TWO INVISIBLE FILES ("TRACK 2.TEXT" & "TRACK 3.TEXT") . PFS WORKS FINE WITH THE "TRACK" FILES VISIBLE AND THE OTHER TWO UNPROTECTED, BUT IF YOU TRY TO COPY THE FILES WITH THE FINDER, THE FILES WILL NOT END UP IN THE SAME PLACE ON THE DISK, SO PFS WON'T RUN ON THE FINDER COPY. (SO PFS IS A "COPYA" OR "DISKCOPY" JOB). OTHER PROBLEMS YOU MIGHT ENCOUNTER ARE MODIFIED FINDER & SYSTEM FILES. MACSLOTS REQUIRES ITS OWN SPECIAL SYSTEM FILE, SO YOU CAN'T PUT ANY SYSTEM-FILE-USING PROGRAMS ON WITH IT. (YOU CAN PUT SOMETHING LIKE ALICE ON WITH IT THOUGH). IF ANY NEW ADVANCES IN MAC-CRACKING EMERGE, I'LL CONTINE THE SAGA....

HAPPY MAC-CRACKING --
THE BYTE
Fort Chappa 203-633-2616

```
==============================================================================
DOCUMENT machine.app
==============================================================================

***************************************
*                              *
*  Black Bag Presents...             *
*                              *
*                              *
*      Cracking Tutorial 001         *
*                              *
*        Machine Language            *
*                              *
*                              *
*           Written By:        *
*     The Intern of Black Bag        *
*            Rev. 1.0          *
*                              *
***************************************
```

  If you desire to do a bit of cracking, it would be a good idea to review (or learn) machine language. You probably have noticed most games are marked with the "B" file type when you "CATALOG" your disk, telling you that they are written in machine language. You may wonder why would some poor masochistic programmer write a game entirely in some bizzare language of ones and zeros. Well, machine language is quick -- very quick. Let's take a look at a sample program for Hi-Res graphics. We can type it in in a minute, but look to see what you're getting into:

```
             Basic
             -----

        10 HGR2 : HCOLOR=3
        20 FOR X=0 TO 279
        30 FOR Y=0 TO 191
        40 HPLOT X,Y
        50 NEXT Y
        60 NEXT X
        70 END

         Machine Language
         ----------------

        10 HGR2 : HCOLOR=3
        20 HPLOT 0,0
        30 CALL 62454
        40 END
```

  Wait! Before you demand a refund, I realize they both look like BASIC, but the second uses a machine language routine built into your apple. It is easier to understand than a whole lot of machine language garbage. Just notice the difference in speed. Although "CALL" is a BASIC command, it is a bridge to machine language. It switches control over to machine language. Before we go any further, let's talk about what machine language is. First, there are different names for machine language. You probably used them interchangably as:

         Binary

        Machine Language
        Assembly

  Luckily, you don't have to worry much about binary. Binary is mainly for machine use only. Binary is the most primitive form and your Apple works mostly in binary because it is based on digital electronics. In digital electronics, you can have either a one or a zero. The one signifies power (voltage), the zero signifies no power. There's not enough room here to go into a deep explanation of a binary logic tree, but I can provide a short summary. A circuit in your computer will check things and receive either a one or a zero. The circuit will travel along a road. When it comes to a fork in the road, it will take the high road if it received a one, but it will take the low road if it has a zero.   At the end of the journey, a specific task would have been completed.

  Each one or zero is a binary digit (or "bit" for short). Obviously, your 6502 can't complete every task if it uses a bit every time it comes to a junction. That is why the 6502 is referred to as an "EIGHT BIT" micro- processorr. In other words, it will take eight bits arranged in a predetermined manner for your 6502 to complete a specific task.

  Four bits in a row make a nybble and two nybbles (or eight bits) make a byte. One byte can be represented as a decimal number from 0 to 255.   The bits in a byte are identified by thier location.   In other words, a byte has 8 bits known as bit 0 through bit 7. The right-most bit is known as bit 0.   To summarize:

```
BITS:  1   1   1   1   1   1   1   1

     !       !   !       !   !       !   !       !   !
     !BIT!BIT!BIT!BIT!BIT!BIT!BIT!BIT!
     ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 !
     +---+---+---+---+---+---+---+---+
     !                 !                 !
     !      UPPER NYBBLE ! LOWER  NYBBLE !
     !                 !                 !
     +---------------+---------------+
     !                             !
     !             ENTIRE BYTE           !
     !                             !
     +-------------------------------+
```

  Before we continue, Let's touch up a bit on your knowledge of base 2 math. Here's an example of the addition of two binary numbers:

```
              1          1
    01              01          01
  + 01       ->>> + 01   ->>>    + 01
  -----     -----        -----
               0         10

 (Step 1)      (Step 2)       (Step 3)
```

  :> Step 1: Start to add.

  :> Step 2: Add first column, there's no "2" in the binary number system, so you must carry the overflow.

  :> Step 3: Continue to add.

Well, you may be confused. It is difficult for many people to convert a binary number into our number system, base 10 or decimal. It's really quite simple if you can visualize that each place in a binary number represents a power of two. If you had a byte like 10000000 (bie 7 = 1), you have the equivilent of two raised to the power of seven. The easiest way to demonstrate this is to use a conversion chart, such as this one:

```
+-----------------------------------+
!     BINARY CONVERSION CHART       !
+-----+---+---+---+---+---+---+---+---+
!BIT  !BIT!BIT!BIT!BIT!BIT!BIT!BIT!BIT!
!PLACE! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 !
+-----+---+---+---+---+---+---+---+---+
! 2'S !   !   !   !   !   !   !   !   !
!     ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 !
!POWER!   !   !   !   !   !   !   !   !
+-----+---+---+---+---+---+---+---+---+
!DEC. !1  !6  !3  !1  !   !   !   !   !
!     ! 2 ! 4 ! 2 ! 6 ! 8 ! 4 ! 2 ! 1 !
!EQUIV!  8!   !   !   !   !   !   !   !
+-----+---+---+---+---+---+---+---+---+
```

```
EXAMPLE:
--------

10001000 = 2^7+2^3 = 1*128+1*8 = 136

 BINARY    POWER OF  MULTIPLY DECIMAL
           TWO
```

Machine language is a general term generally meaning a low level language which is heavily dependant on the machine. For example, machine language on an IBM is different than the machine language on an Apple. Machine language is a very broad area. There are several different levels. Organized from lnwest to highest, they are:

Translated
Interactive
Mini-Assembler
Macro-Assembler

The first refers to poking your program in from another language. This often used from within BASIC. The reason this is used is to make a program "self-contained," or to extend the limit of a machine. On a Vic-20, machine language is not supported. In order to do any machine language on a Vic-20, without purchasing an expansion module, you would have to "POKE" your program into memory from BASIC and then "CALL" it.

The second, the interactive mode, usually is accomplished by a "monitor." This is not a CRT, it is a small program which contains most of the routines for running your machine (keyboard input, character output, sound, and other things) and also contains a small command processor which allows you to write machine language programs and list them. When you type "CALL-151" on an Apple, you enter the "monitor" and can begin coding programs. The major drawback of the interactive and the translated method, is that you must know the numbers which correspond to commands. Machine language is much harder than BASIC simply because line numbers, variables, data, and commands are all represented by the same kind of numbers. It is very similar to Chinese. In Chinese, the same word may be an obscenity or a complement.

A mini-assembler allows you to write machine language using words for commands rather than numbers. They can use line numbers (actually memory locations) such as the "mini-assembler" located within Integer BASIC, or "labels." A label allows you to name a subroutine and not have to worry about mathematically calling it. In BASIC, it would be similar to typing "GOSUB HOUSEKEEPING"" instead of "GOSUB 100." Most mini-assemblers also allow the use of variables or a name for a specific memory location that you will use for storage. When you use Apple's built-in monitor, you must always know the actual location, or a number, that you will use as a variable. Thus, mini-assemblers allow you to write programs very quickly because you need not memorize a bunch of numbers and sequences.

A Macro-Assembler is the best. In BASIC, most of us have at least one typical subroutine that we use over and over again even for different programs. Each time we must re-enter it into memory. In a macro assembler, we can write this subroutine once, name it, and save it. Then, when we write our program, we need only to call it by name, and the machine will automatically insert your subroutine. Usually macro assemblers have psuedo-code options, which allow you to control the translation of text to the actual program. They can control exactly what microprocessor is installed, or how much memory is available.

The last two are the easiest to use, you would simply write your machine language program in a text editor and compile them, or "assemble" them. They are very effecient because if you need to insert 1 or 100 lines of code, they will automatically refigure the program just by compiling the program again. The first two are good for short and quick applications. When you just want to test a small subroutine, or impress a your computer teacher. It is best to begin he hard way, with the translated and interactive methods, because all four forms are bound by the same rules, and any mistakes you my have made will be easy to see and even easier to change. But, if you want to begin, you must know a whole new counting system. The hexadecimal counting system, or base 16.

Hexadecimal breaks up a byte into two characters. One character represents the upper nybble, the other represents the lower hybble. There is a problem though, decimal has only 10 symbols, "0" to "9", and hexadecimal system needs 16 distinct characters. To solve the problem, the symbols "A" through "F" are used to represent 10 to 15.

NYBBLE	DECIMAL	HEX
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A

```
      !  1011  !   11    !  B  !
      !  1100  !   12    !  C  !
      !  1101  !   13    !  D  !
      !  1110  !   14    !  E  !
      !  1111  !   15    !  F  !
      !        !   !     !
      +--------+--------+-----+
```

The conversion of a number to hex just takes practice, but becomes relatively easy if you take a number and break it down to the binary level, where conversion to hex is as easy as pie.

Just as nybbles may be organized into groups of two, so may bytes. Two bytes together usually represent a number. The number may be expressed in two different ways. Let us use the example of the number FF00:

```
         A Two Byte Number
         -----------------

    F F 0 0 =     0 0   F F
    \ / \ /       \ /   \ /
     !       !           !     !
     Hi  Lo        Lo    Hi
                  Byte  Byte
```

When the two bytes are written together, the high byte is written first. When the two bytes are seperated, the low byte is written first. This is one difference you will notice between the different levels of machine language. An assembler will translate the standard format of two bytes written together to the machine format of "lo byte first." You can blame the designers of computers for the strange splitting of numbers, but it really makes much more sense to the computer in the lo-hi format.

One term you should know is "word." The word is simply two bytes. The numbers in the paragraph above are one word long. Sixteen bit computers look at memory in increments of words, while 8 bit computers look at memory in increments of bytes.

To distinguish between memory locations, values, and the different forms of number systems, some symbols are used in addition to numbers:

```
 % = Binary Location Value
#% = Binary Number
   = Decimal Location (no symbol)
 # = Decimal Number
 $ = Hexadecimal Location Value
#$ = hexadecimal Number
```

Following the symbols would be the digits. Thus, the hexadecimal location 10 would be written $10. The binary number 0110 would be written #%0110. Again, this does not pertain to the monitor or interactive forms of writing machine language, only the forms using an assembler. In an attempt to keep things simple, I have ommited the symbol notations.

This is still not true machine language, since that is a nazt. If you did not understand much of the tutorial, do not worry. For now, just understand the hexadecimal numbering system.

For the next set of docs, try to find a copy of the "DOS Toolkit," and look over the following examples. Until then, good luck!

The Intern
of Black Bag

Hex		Decimal		Binary
FF	=	255	=	1111 1111
D0	=	208	=	1101 0000
B7	=	183	=	1011 0111
A0	=	160	=	1010 0000
74	=	116	=	0111 0100
37	=	55	=	0011 0111
13	=	19	=	0000 1101

```
==============================================================================
DOCUMENT machinel.app
==============================================================================
```

Filename: M/L Part I

```
***************************************
*                             *
*                             *
*   MACHINE LANGUAGE TUTORIAL DISK    *
*                             *
*      WRITTEN BY DR. FIRMWARE            *
*                             *
*                             *
***************************************
```

The aim of this disk is for you the reader to understand machine language to an extent so that you can program fully in machine language (ml).

PART I
======

The fundamentals.

The first part of the course is number bases. if you undestand binary and hexadecimal numbers and conversion between these and decimal, you can skip to the next section.

Binary: Base two.

Number bases are what we are dealing with here. The number base that we normally use in everday life is decimal. 'Decimal' comes from latin where it meant ten. We have ten digits, 0,1,2,3,4,5,6,7,8, and 9, which are combined in various ways to produced other numbers. It is understood that the number '345' means 3x100+4x10+5x1. The right-most digit has the least significance, while the left-most has the most significance. FrgiO∑∑«∑∑O∑∑
a©8∑\P$B∑∑ !®   §¥fl$∑¤´`∑.¿#∂Îƒ]∑∑∑4∑®`\G±.∑ »

¢®

```
∑`∑∑∑´Ta∑  `hQ∑∑∑Ê∑∑Ó•a+¥phPhRs{∑
∑∑c√Ks9¥C+¨)3∑s#
k+s¥
c™a∑);ca{s¨¥®∑∞¥C)
¨)¥∑ya{•Ks
®¡as∑k+§hS¨√¨¥+iq3K®¨∞¥C+®)
®)¥∑y#K;K¥™a∑a
s!∑q¨ya¥C)®K;C∞kk{¨∞#K;K∞C
™¥C(hSc+
¨∞¨K;sK3K
s)
s!¥C)c+3∞kk{¨∞a¥C)k{¨∞¨K;sK3K
s)aS∑¨∞cK[)KphS#+Kk
aqs{∑a¥C)s∑k+®™k∑c¥K∑cK+!∑K¥A¥C)#K;Hts will be successive powers
```
of two. 2~0=1, 2~1=2, 2~2=4, 2~3=8, etc. We now have the basics down, so we'll take a number, such as '1001101', and find it's decimal value.

To start, we'll take the right-most digit and find out what it is multiplied with. Since it's the right- most digit, it's multiplied with two to the power of zero. 1x2~0=1. Now, repeat the process, this time with the second right most digit, which is a 0. 0x2~1=0. Continueing produces: 1x2~2=4, 1x2~3=8, 0x2~4=0, 0x2~5=0, and 1x2~6=64. Summing the results, 1+0+4+8+0+0+64=77. So 77 is the decimal value of the binary number 1001101.

If you want to practice some, just make strings of 0's and 1's and do what we did above.

Conversion from decimal to binary is a little more complex. Suppose we take a decimal number, 35. To convert, we do a series of steps.

1> Divide the number by two, and put the remainder aside.

2> Replace the dividend with the quotient.

3> Repeat step 1 & 2 until the number reaches zero.

4> Take the remainders and place them in a row, the first is right-most, the last is left-most.

And that's it. To demostrate, we'll convert 35 to binary.

```
       0 R=1 -------
     ---        !
   2)  1 R=0          !
     ---        !
   2)  2 R=0          !
     ---        v
   2)  4 R=0          100011
     ---           ~
   2)  8 R=1             !
     ---           !
   2) 17 R=1 ------------
     ---
   2) 35
```

There. Quite simple. The diagram would look somewhat better on paper, but this will have to do in the mean while.

Hexadecimal

'Hex', as it is affectionately called by in most computerese dialects, is nothing more than a base sixteen number system. Let's go through some basics.

It has 16 digits. These digits are the numbers 0-9, and the letters A-F. The reason why the letters are included is because there aren't enough numbers. Let's take a number, $4A. Note that when you see a '$' infront of a number, it denotes that the number is a hex number. $4A means 4x16~1+10x16~0. The letters are the numbers from 10-15, A being 10, B is 11, C=12, etc.

Conversion to decimal is exactly the same as for binary. To demostrate we'll convert 10234 to hex.

```
          0
       ------
    16)     2  R=7  ----
       ------          v
    16)    39  R=15    7FA
       ------          ~
    16)   639  R=10 -----/
       ------
    16) 10234
```

There we are! 10234 is $7FA.

One interesting fact: since 16=2~4, then a 4 digit binary number is equal to 1 hex digit, i.e. 1111=$F, 1010=$A, etc. This makes binary to hex, and vice versa, conversion very easy. For example, the number $3A0 in binary is

```
  0011 1010 0000.
   ~  ~    ~
   !  !    $0
   !  $a
   $3
```

This ends the discussion on number bases and now the reader should be aquianted with binary and hex and what they mean. Digital is really only a binary digit. In other words, a 1 or a 0. These are digital computers handle, strings upon strigs of bits. Unfortunately, bits are very combersome, because even the charcters that you see require 8 bits each. The screen size is 40x24, and that adds up to 7680! bits!

A more convinient form are two digit hex numbers. A two digit hex number represents 8 bits in only two digits. A more common name for this compact unit is a byte.

You might know that your computer has 64K RAM. The K represents 1,024 bytes. So this means that your computer has 65,536 bytes of RAM memory. 65,536 can be expresses more conviniently as 2~16. This is important for reasons that we'll discuss a little later.

Well, there we are! Now that we have some basics down, we can get to some machine language.

=-=

```
***************************************
*                              *
*                              *
*   MACHINE LANGUAGE TUTORIAL DISK    *
*                              *
*      WRITTEN BY DR. FIRMWARE            *
*                              *
*                              *
***************************************
```

PART II
=======

Machine language command structure.

Even though this sounds complicated, the structure of machine language commands is quite simple. The command is one to three bytes long and consists of two sections, the operator and the arguement. The operator is always one byte long and the arguement is either zero, one or two bytes long. If the arguement is zero bytes long, then it is said that there is no arguement for that command.

The accumulator

The accumulator is the primary register in the 6502 microprocessor. It is an 8 bit register, which means that it can handle only eight bits at a time or the numbers from zero to 255.

To put numbers into the accumulator, we use a command called LDA which stands for LoaD Accumulator. This command takes the value generated by the arguemant and places it into the accumulator.

Addressing modes

Addressing modes are very important. These tell the computer how to deal with the arguement that it recieves. We will only be dealing with two modes for the present, immeadiate, and absolute.

In immeadiate addressing mode, the LDA command load the accummulator with the actual value of the arguement. Suppose that we wanted to load the value $6F into the accumulator. We would do this by telling the microprocessor to 'LDA #$6F'. That is assembly language. In actual fact, the code used by the microprocessor would represent it as '$A9 $6F'. The $A9 tells the microprocessor that you want to load the accumulator in immeadiate addressing mode. The $6F is the arguement and is treated as described above. So then, the number $6F is put directly into the accumulator.

The LDA command in immeadiate addressing mode is two bytes long. The first byte being the operator ($A9) and the second being the arguement.

Memory locations.

The Apple computer has 2~16 memory locations. Each memory location is 8 bits large. Each memory location can be referenced by a 4 digit hex number. A four digit hex number is 2 bytes long and can be cut in half into two separate bytes.

The byte on the left is more significant than the one on the right, so the one on the left is called the Most Significant Byte (MSB) and the one on the right is the Least Significant Byte (LSB).

In absolute addressing mode, the LDA command takes the arguement as an address and then takes the value held in that address and transfers it to the accumulator. The arguement is two bytes long and it forms the address LSB first and MSB second. The address is in effect backwards.

Say you wanted to load the accumulator with whatever was in location $456D. The operator is $AD, this is followed by the LSB which is $6D, and finally the MSB, $45.

Storing the accumulator.

To move the contents of the accumulator to some other memory location, we use the command STA, which stands for STore Accumulator.

The STA command has an absolute addressing mode. The hex operator is $8D and it is followed by the LSB and MSB, in that order. After the command is executed, the accummulator still contains the value.

Now we can make a tiny program to store the value $8D into location $2000. First, we have to load it into the accumulator. To do this, we'll load the $8D into the accumulator through the LDA immeadiate command. So, then we'll store the accumulator into $2000 while it contains our value using the STA absolute command.

In assembly language, our program looks like this:

```
LDA #$8D
STA $2000
RTS
```

Note: the '#' indicates that the command is in immediate addressing mode. The RTS is going to be used as a general 'end' command for now, until I can explain it's actual usage.

This assembly language version is not understandable by the microprocessor. It has to be translated into hex codes. This translation is normally done by an assenbly program, but since this is a short program, we'll do it by hand.

We are going to put this program at location $300-$306. This area can be used for short programs as $300-$3b0 is free memory space. An extended memory map will be included in a later edition.

```
LDA #$8d   -->      $A9 8D
STA $2000  -->      $8D 00 20
RTS      -->        $60
```

```
hex location         contents
---------------------------
 $300          $A9
 $301          $8D
 $302          $8D
 $303          $00
 $304          $20
 $305           $60
```

The program can be entered into memory using the BASIC POKE command. $300 is equal to 768 and the rest of the hex numbers you should be able to convert into decimal yourselves.

This concludes PART II of the series. Coming next: X and Y registers.

=-=

```
********************************∑
**P∑P-I∑sÌ∑∑I…÷∑!÷®°»∞∑∑ƒ∑•∑Ã∑∑∑¡…∑!(¤A09
¥…L∑Ã1=∑?∞®∑∑∑»$Ò@; OQ¡∑< \ƒ!Ó®®A&'7-
And ™≠wue        £:0(  Q
∑∑∏∞∑····ØØØØØØ∑¤‹∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑¤‹¤‹¤‹<
      Ø$$$¤‹∑∑∑∑∑∑∑∑¤‹¤‹ØØØØ
      ∑¤‹∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑∑¤‹¤‹Ø       Ø     ØØ          Ø     Ø
            ØØØ∑∑£∑Ø                    ∑¤‹ØØØØØ
      ØØØØ                   ∑ØØØØØ          ster,
```
we use the LDX command. This command works in both immediate and absolute addressing mode. The STX command stores the X register in an address the same way as the STA command does, operator, LSB, and then the MSB.

The Y register it affected by the LDY and STY commands.

Absolute indexed addressing mode.

The X and Y registers are also called 'index' registers. This is because they can be used to index the accumulator to generate a 'flexible' address.

When one uses absolute addresing in loading the accumulator, then the program is using a 'fixed' address, in that the address reference remains the same all the time. This is desirable, but there are instances which require a certain byte within a range of memory depending on other factors. Providing the range is small (2-4 locations), one can do it with some branching commands, but if it goes beyond 10 locations, this can become a nightmare. Indexing provides a very simple solution. When using the LDA command in absolute indexed addressing mode, the X or Y register (the register that will be used is specified by the operator, there is one opertaor for each register ($BD for indexing with X and $B9 for Y.) For the discussion we'll assume that the X register is being used to index the accumulator) is added to the value of LSB the address in the arguement (carry is considered) and then the accumulator is loaded with the contents of the resulting address.

For example, suppose the X register holds $50 and the program executed a command LDA $2000,x. The LDA is to tell us that we want to load the accumulator with a number. The '$2000,x' tells us that we'll be using absolute indexed addressing and that the indexing register is the X register. This is what happens in the circuitry: We take the 'base' address, $2000, and add the value of the X register to the LSB. This gives us $2050. The contents of that location is then copied into the accumultor.

As you can see, the X register is used to 'offset' the accumulator and it produces various addresses as the value in the X changes. Also since the X register can only hold the numbers from 0-255, then you can only offset by that much.

Storing in absolute indexed is exactly like loading. The same principle applies except that instead of transfering a byte from memory to the accumulator, you're transfering a byte from the accumulator to memory.

There also exists an LDY absolute indexed X, that is, load Y absolute, but add X to the LSB of the address, and an LDX absolute indexed Y. These are useful when the accumulator is busy holding some important data. Unfortunately, you cannot store either X or Y register indexed in absolute addressing mode.

Indexing is quite useful at times. However, the usefulness will be exposed to a much greater depth in the next installment when we cover branching. I will leave for now with some notes on memory orginization.

64K....

That is the total directly addressable memory that your apple computer has. Now wait, I know, you've got a 128K card sitting in slot 0, that's fine and dandy, but it's not all availble at once. For now, just consider a basic 48K system, and later I'll tell you how the ramcards work.

64K is equal to $10000, or $0 through $FFFF, memory locations. (In the latter case, location $10000 is the same as location $0.) This is subdivided into 256 $100 byte pages. Page zero would be locations $0000 through $00FF, page one would be $0100-$01FF, etc...

Zero page (page zero, if you prefer) is special. In this range of memory, there exists many pointers, flags and other stuff that is very crucial to the smooth operation of BASIC. One of the reasons why is because the MSB of all the locations is zero.

Page one is reserved for the hardware stack. You'll know all about it in a short while.

Page two is reserved by the BASIC input routine. We'll cover that very soon, too.

Page three is mostly free programmimg space. You might notice that most of the shorter programs that we will write will be located in this area.

Pages four to seven ($400-$7FF) are the primary text screen. (Yes, there are two text screens.)

From $800 to $95FF is BASIC programming space (or M.L. space if you want it).

Under DOS 3.3, from $9600 to $BFFF is taken by DOS.

From $C000-$CFFF is periphreal softswitches and PROM programming space.

And finally, $D000 to $FFFF is taken up by ROM.

Your basic Apple memory map. Of course, we'll be expanding on it greatly, but for now, this is it.

=======================================

```
=============================================================================
DOCUMENT macteam.app
=============================================================================
```

```
-----------------
MACTEAM CONFERENCE
-----------------
```

[Below are highlights of an online conference conducted recently in MAUG, the Apple users section over on CompuServe. Special guests were Cary Clark, Guy Kawasaki and Dan Cochran, three key Apple managers involved in Macintosh software development and technical support. They answered questions on writing commercial software for the Mac, on upcoming Apple products and on other topics of interest to Mac software developers. The conference took place on 9/9/84, the day prior to the introduction of the 512K Macintosh. The questions were asked by various MA UG members.]

---<*****>---

Q: Many user groups have purchased " Inside Macintosh" and the Software Supplement for their members. Can any of the software in those packages be freely distributed to user group members, in particular the Resource Mover and Font Editor?

A: Those last two are okay for you to informally distribute. We do sell the Supplement to ensure that everyone receives updates to documentation and software. It is in your best interest to purchase the Supplement. Also, you should be forewarned that the Font Editor and Resource Mover have many bugs and are hard to use. They will both be replaced by the Resource Editor, part of a future software supplement.

Q: I would like to know what is necessary to get certification for development.

A: The Apple Certified Developer Program is administered by the Developer Relations Group. We are looking for a serious commitment to commercial development of products to enhance the saleability of our hardware. To get an application, please write to Developer Relations, Apple Computer Inc., 20525 Mariani - MS 23AF, Cuper tino CA 95014.

Q: Is there a license fee for developers to pay to Apple on products?

A: The Finder, desk accessories and system can be licensed for unlimited use for under $100 per year per product. Such a deal. For licensing, please contact Toni Tommacci at 408- 996-1010.

Q: What's the holdup on the Lisa 1 upgrades? When will the free one and the Lisa 2/10 be readily availab le?

A: If you are an Apple certified developer and are having trouble getting a Lisa 2/5 upgrade, please contact Kathy Schlein at the 20525 address, MS 2T. The Lisa 2/10 upgrades are not readily available.

Q: I am working on some desk accessories. The problem is that they are too big, around 16K or more. Is a 512K Mac going to alleviate any problems I am having with the 'opendeskacc' call NOT preventing a bomb?

A: When you're a desk accessory, you 're a guest in someone else's house so you gotta be inconspicuous. 16K is toooo big. As a point of reference, the control panel, which is the largest desk accessory, fits into 6K, with all its pictures. The 512K really won't help you, since people writing applications for the 512K Mac will still expect small desk accessories, and there will still be the large installed base of 128K Macs.

Q: Is floating-point (SANE) stuff going to be transparent in any future languages? It is a real pain to use it when formulas are complex.

A: Yes, it will be transparent (in-line) in future development languages, including the Lisa Pascal compiler by the end of the year. SANE is an insanely great package, IEEE-certified, better than most mainframe pack- ages, so now we're just making it easier to use. You might say that it's the numerics package for the rest of us!

Q: I've heard rumors that the Mac ROM has already gone through several revisions since the Mac started shipping. Any truth to that?

A: The ROM has not been revised since Macintosh shipped. The System Disk was revised once, on May 7th.

Q: Whatever happened to CoreEdit? "Inside Mac" now says the documen- tation for it doesn't even exist, and yet I HAVE seen CoreEdit docs in an earlier version of IM.

A: CoreEdit is only the assembly-language part of MacWrite and has no system support. It is only a piece of an application. Way back before the Macintosh was born, CoreEdit was going to be part of the ROM. That's why the documentation was written back in March of '83. But nothing has been done since then. The CoreEdit of today would allow you to write only one application: MacWrite. And that's already been done.

Q: I am a non-certified owner of Inside Mac and the Software Supplement. But I would really like to know the internal details of the MacWrite file format. Any way I can get that information?

A: The problem with that is that there are already two formats in exis- tence, with more to come. If you write any software around it, it will only work for a limited period of time and will not be compatible with other programs. MacWrite will continue to change too frequently for you to benefit from the document format. Instead, you should be compatible with the TEXT format, which is defined as 'vanilla' text separating paragraphs with carriage-returns.

Q: Do you plan to support Macintosh software development on machines other than the Lisa, like the Apple // or the Mac itself? There's already an excellent 68000 cross-assembler for the Apple // from S-C Software.

A: Native Macintosh development environments are very important to Apple. The 68000 Development System will be released in October, and we are working on a 512K Mac-based native development environment. This environment will support assembler, Pascal and C in a common support environment. There are also a lot of third-party native development environments popping up.

Q: Can you tell me what percentage of Macs have gone to Fortune 1000 companies? What would increase penetration of this market for Apple?

A: I'd guess that 5 to 10 percent of Macintosh sales are going to the Fortune 1000 market. We are expecting to increase our presence in this market with Macintosh office products such as Applebus, laser printers, file servers, etc.

Q: What is the status of Applebus, in particular the status of an "Apple" hard disk or networking setup for Macs?

A: You can get the complete specifications for Applebus by mailing $75 to Apple Computer Inc., 476 Saratoga Ave. - Suite 621, San Jose CA 95129. Please mention that you want "Inside Applebus". To get a 10-page quicky summary, mail a note to Apple Comput er Inc., 10455 Bandley Drive - MS 2T, Cupertino CA 95014, Attention: Steve Hoyt.

Q: Several questions. First, is there any possibility of a multi-tasking version of the Finder? Second, when can we expect 15-inch Imagewriter support? And finally, how far along is Lotus's product and will it be available when 512K Macs are?

A: Several answers. The Finder will run the calculator, the clock and the control panel at the same time right now. Really, it is not a Finder restriction, it is a ROM restriction . 15-inch Imagewriter support is in beta-test and should show up by the end of September. Lotus will be available in the mid-1st quarter of 1985.

Q: Is there going to be a double-sided drive?

A: The current ROM supports double-sided drives. Sony engineers are working on it.

Q: Can you tell us what support MacBasic and MacPascal will have for ROM routines?

A: MacPascal will support all of QuickDraw and a few of the most useful ROM routines. A future version will support the entire ROM. MacBasic will support about 250 of the ROM routines.

Q: Why the delay with MacTerminal?

A: MacTerminal is now in production, thanks to its authors, Mike Boich and Martin Haeberli. Did you know that Certified Developers can license the source codes for $2500 if they add value or customize it?

Q: What are some of the reasons you're hearing from software developers for the delays in release of their programs? Are many waiting for the 512K Macs?

A: In general, software developers are not waiting for the 512K release. Only Lotus is specifically targeting that version of the Mac. The delays are caused by the "learning hump" for writing Macintosh applications. There's just a lot to learn about Macintosh, so the second application is much easier and faster to write. Ask Bob Hardy of Penguin or Bert "BugBuster" Porter of Blue Chip. On the other hand, look at Filevision and Dollars & Sense. They wrote those applications in a relatively short period of time and they are in sanely great.

Q: What are the chances of Apple using the new Motorola 68020 micro-processor in future versions of the Mac?

A: The 68020 is a neat chip. The Macintosh architecture is processor-independent.

Q: I get the impression that there is a whole family of Macs or other permutations in the wings. Can you comment on what versions are on the drawing boards or even possibly on the assembly lines?

A: The next permutation of Macintosh will be the 512K version. We really cannot comment on future versions except to say that we are totally committed to the Macintosh a rchitecture.

Q: What's the bozo bit?

A: The bozo bit is a crude form of copy protection, hence its name.

Q: What would Apple prefer to see concerning software copy protection and software pricing by outside developers?

A: Copy protection should be invisible to the end-user. As for pricing, be sure your prices exceed your fully allocated costs.

```
=============================================================================
DOCUMENT memory.txt
=============================================================================

                  APPLE  CALL, PEEK, POKE LIST

-----------------------------------------------------------------------------
CALL  -144     SCAN THE INPUT BUFFER
CALL  -151     ENTER THE MONITOR NORMALLY
CALL  -155     ENTER THE MONITOR & SOUND BELL
CALL  -167     ENTER MONITOR AND RESET
CALL  -198     RING BELL (SIMULATE CONTROL G)
CALL  -211     PRINT "ERR" AND RING BELL
CALL  -259     READ FROM TAPE
CALL  -310     WRITE TO TAPE
CALL  -321     DISPLAYS A, S, Y, P, & S REGISTERS
CALL  -380     SET NORMAL VIDEO MODE
CALL  -384     SET INVERSE VIDEO MODE
CALL  -415     DISASSEMBLE 20 INSTRUCTIONS
CALL  -458     VERIFY (COMPARE & LIST DIFFERENCES)

CALL  -468     MEMORY MOVE AFTER POKING   60,61 OLD START - 62,63 OLD END
                                     64,65 NEW END   - 66,67 NEW STAR

CALL  -484     MOVE
CALL  -517     DISPLAY CHARACTER & UPDATE SCREEN LOCATION
CALL  -531     DISPLAY CHARACTER, MASK CONTROL CHAR., & SAVE 7 REG. & ACCU
CALL  -550     DISPLAY HEX VALUE OF A-REGISTER (ACCUMULATOR)
CALL  -656     RING BELL AND WAIT FOR A CARRIAGE RETURN

CALL  -657     GET LINE OF INPUT, NO PROMPT, NO L/F, & WAIT(COMMA,COLON OK
CALL  -662     GET LINE OF INPUT, WITH PROMPT, NO L/F, & WAIT
CALL  -665     GET LINE OF INPUT, WITH PROMPT, LINE FEED, & WAIT
THE ABOVE 3 CALLS (-657, -662, -665) REFER TO THE INPUT BUFFER FROM 512-767

CALL  -715     GET CHARACTER
CALL  -756     WAIT FOR KEY PRESS
CALL  -856     TIME DELAY (POKE 69,XX TO SET TIME OF DELAY)
CALL  -868     CLEARS CURSOR LINE FROM CURSOR TO END OF LINE
CALL  -912     SCROLLS TEXT UP 1 LINE
CALL  -922     LINE FEED
CALL  -936     CLEAR SCREEN (HOME)
CALL  -958     CLEAR SCREEN FROM CURSOR TO BOTTOM OF SCREEN
CALL  -998     MOVES CURSOR UP 1 LINE
CALL  -1008    MOVES CURSOR BACKWARD 1 SPACE
CALL  -1024    DISPLAY CHARACTER ONLY
CALL  -1036    MOVES CURSOR FORWARD 1 SPACE
CALL  -1063    SEND BELL TO CURRENT OUTPUT DEVICE
CALL  -1216    TEXT & GRAPHICS MODE
CALL  -1233    MOVE CURSOR TO BOTTOM OF SCREEN
CALL  -1321    CONTROL E
CALL  -1717    MOVES CURSOR DOWN 5 LINES
CALL  -1840    DISASSEMBLE 1 INSTRUCTION
CALL  -1953    CHANGE COLOR BY +3
CALL  -1994    CLEAR LO-RES SCREEN (TOP 40 LINES)
CALL  -1998    CLEAR GRAPHIC SCREEN (LO-RES)
CALL  -2007    VERTICAL LINE
CALL  -2023    HORIZONTAL LINE
CALL  -2458    ENTER MINI ASSEMBLER
CALL  -3100    TURNS ON HIRES PAGE 1, WITHOUT CLEARING IT
CALL  -3776    SAVE INTEGER
CALL  -3973    LOAD INTEGER
CALL  -6090    RUN INTEGER
CALL  -8117    LIST INTEGER
CALL  -8189    ENTER BASIC & CONTINUE
CALL  -8192    ENTER BASIC AND RESET (INTEGER BASIC KILL)
CALL  -16303         TEXT MODE
CALL  -16304         GRAPHICS MODE
CALL  -16336         TOGGLE SPEAKER
CALL   42350         CATALOGS DISK
CALL   54915         CLEANS STACK, CLEARS THE "OUT OF MEMORY" ERROR
CALL   64166         INITIATES A COLD START (BOOT OF THE DISK)
CALL   64246         BRAND NEW-YOU FIGURE IT OUT

CALL   64367         SCANS MEMORY LOC 1010 & 1011 & POKES VALUE INTO LOCATIONS
               1012 THAT IS EQUAL TO (PEEK(1011)-165)

------------------------------------------------------------------------------
PEEK   33      WIDTH OF TEXT WINDOW (1-40)
PEEK   34      TOP EDGE OF TEXT WINDOW (0-22)
PEEK   35      BOTTOM OF TEXT WINDOW (1-24)
PEEK   36      HORIZONTAL CURSOR POSITION (0-39)
PEEK   37      VERTICAL CURSOR POSITION (0-23)
PEEK   43      BOOT SLOT X 16 (AFTER BOOT)
PEEK   44      END POINT OF LAST HLIN, VLIN, OR PLOT
PEEK   48      LO-RES COLOR VALUE X 17

PEEK   50      TEXT OUTPUT FORMAT: 63=INVERSE   255=NORMAL
               127=FLASH ( WITH PEEK 243 SET TO 64)

PEEK   51      PROMPT CHARACTER
PEEK   74,75         LOMEM ADDRESS (INT)
PEEK   76,77         HIMEM ADDRESS (INT)
PEEK   103,104       FP PROGRAM STARTING ADDRESS
PEEK   104     IF 8 IS RETURNED, THEN FP IS IN ROM
PEEK   105,106       FP VARIABLE SPACE STARTING ADDRESS
PEEK   107,108       FP ARRAY STARTING ADDRESS
PEEK   109,110       FP END OF NUMERIC STORAGE ADDRESS
PEEK   111,112       FP STRING STORAGE STARTING ADDRESS
PEEK   115,116       FP HIMEM ADDRESS
PEEK   117,118       FP LINE NUMBER BEING EXECUTED
PEEK   119,120       FP LINE WHERE PROGRAM STOPPED
PEEK   121,122       FP LINE BEING EXECUTED ADDRESS
PEEK   123,124       LINE WHERE DATA BEING READ
PEEK   125,126       DATA LOCATION ADDRESS
PEEK   127,128       INPUT OR DATA ADDRESS
PEEK   129,130       FP LAST USED VARIABLE NAME
PEEK   131,132       FP LAST USED VARIABLE ADDRESS
PEEK   175,176       FP END OF PROGRAM ADDRESS
PEEK   202,203       INT PROGRAM STARTING ADDRESS
PEEK   204,205       INT END OF VARIABLE STORAGE
PEEK   214     FP RUN FLAG (AUTO-RUN IF >127)
PEEK   216     ONERR FLAG (>127 IF ONERR IS ACTIVE)
PEEK   218,219       LINE WHERE ONERR OCCURED
PEEK   222     ONERR ERROR CODE
```

PEEK	224,225	X-COORDINATE OF LAST HPLOT
PEEK	226	Y-COORDINATE OF LAST HPLOT
PEEK	228	HCOLOR VALUE 0=0 85=2 128=4 213=6
		42=1 127=3 170=5 255=7
PEEK	230	HI-RES PLOTING PAGE (32=PAGE 1 64=PAGE 2 96=PAGE 3)
PEEK	231	SCALE VALUE
PEEK	232,233	SHAPE TABLE STARTING ADDRESS
PEEK	234	HI-RES COLLISION COUNTER
PEEK	241	256 MINUS SPEED VALUE
PEEK	243	FLASH MASK (64=FLASH WHEN PEEK 50 SET TO 127)
PEEK	249	ROT VLAUE
PEEK	976-978	DOS RE-ENTRY VECTOR
PEEK	1010-1012	RESET VECTOR
PEEK	1013-1015	AMPERSAND (&) VECTOR
PEEK	1016-1018	CONTROL-Y VECTOR
PEEK	43140-43271	DOS COMMAND TABLE
PEEK	43378-43582	DOS ERROR MESSAGE TABLE
PEEK	43607	MAXFILES VALUE
PEEK	43616,46617	LENGTH OF LAST BLOAD
PEEK	43624	DRIVE NUMBER
PEEK	43626	SLOT NUMBER
PEEK	43634,43635	STARTING ADDRESS OF LAST BLOAD
PEEK	43697	MAXFILES DEFAULT VALUE
PEEK	43698	DOS COMMAND CHARACTER
PEEK	43702	BASIC FLAG (0=INT 64=FP ROM 128=FP RAM)
PEEK	44033	CATALOG TRACK NUMBER (17 IS STANDARD)
PEEK	44567	NUMBER OF CHARACTERS MINUS 1 IN CATALOG FILE NAMES
PEEK	44611	NUMBER OF DIGITS MINUS 1 IN SECTOR AND VOLUME NUMBERS
PEEK	45991-45998	FILE-TYPE CODE TABLE
PEEK	45999-46010	DISK VOLUME HEADING
PEEK	46017	DISK VOLUME NUMBER
PEEK	46064	NUMBER OF SECTORS (13=DOS 3.2 16=DOS 3.3)
PEEK	49152	READ KEYBOARD (IF >127 THEN KEY HAS BEEN PRESSED
PEEK	49200	TOGGLE SPEAKER (CLICK)
PEEK	49248	CASSETTE INPUT (>127=BINARY 1, <128=BINARY 0)
PEEK	49249	PADDLE 0 BUTTON (>127 IF BUTTON PRESSED)
PEEK	49250	PADDLE 1 BUTTON (>127 IF BUTTON PRESSGD)
PEEK	49251	PADDLE 2 BUTTON (>127 IF BUTTON PRESSED)
PEEK	49252	READ GAME PADDLE 0 (0-255)
PEEK	49253	READ GAME PADDLE 1 (0-255)
PEEK	49254	READ GAME PADDLE 2 (0-255)
PEEK	49255	READ GAME PADDLE 3 (0-255)
PEEK	49408	READ SLOT 1
PEEK	49664	READ SLOT 2
PEEK	49920	READ SLOT 3
PEEK	50176	READ SLOT 4
PEEK	50432	READ SLOT 5
PEEK	50688	READ SLOT 6 (162=DISK CONROLLOR CARD)
PEEK	50944	READ SLOT 7
PEEK	64899	INDICATES WHICH COMPUTER YOU'RE USING
		223=APPLE II OR II+, 234=FRANKLIN ACE OR ?, 255=APPLE IIE
POKE	33,33	SCRUNCH LISTING AND REMOVE SPACES IN QUOTE STATEMENTS
POKE	36,X	USE AS PRINTER TAB (X=TAB - 1)
POKE	50,128	MAKES ALL OUTPUT TO THE SCREEN INVISIBLE
POKE	50,RANDOM	SCRAMBLES OUTPUT TO SCREEN
POKE	51,0	DEFEATS "NOT DIRECT COMMAND", SOMETIMES DOESN'T WORK

```
POKE   82,128        MAKE CASETTE PROGRAM AUTO-RUN WHEN LOADED
POKE   214,255       SETS RUN FLAG IN FP & ANY KEY STROKES WILL RUN DISK
      PROGRA
POKE   216,0         CANCEL ONERR FLAG

POKE   1010,3        SETS THE RESET VECTOR TO INITIATE
POKE   1011,150    A COLD START (BOOT)

POKE   1010,102    MAKE
POKE   1011,213    RESET
POKE   1012,112    RUN

POKE   1014,165    SETS THE AMPERSAND (&) VECTOR
POKE   1015,214    TO LIST YOUR PROGRAM

POKE   1014,110    SETS THE AMPERSAND (&) VECTOR
POKE   1015,165    TO CATALOG A DISK

POKE   1912+SLOT,1 ON APPLE PARALLEL CARD (WITH P1-02 PROM) WILL ENABLE L/F'S
POKE   1912+SLOT,0 ON APPLE PARALLEL CARD (WITH P1-02 PROM) WILL ENABLE L/F'S

POKE   2049,1        THIS WILL CAUSE THE FIRST LINE OF PROGRAM TO LIST REPEATEDLY
POKE   40514,20    ALLOWS TEXT FILE GREETING PROGRAM
POKE   40514,52    ALLOWS BINARY FILE GREETING PROGRAM

POKE   40993,24    THIS ALLOWS
POKE   40994,234   DISK COMMANDS IN
POKE   40995,234   THE DIRECT MODE

POKE   42319,96    DISABLES THE INIT COMMAND

POKE   42768,234   CANCEL ALL
POKE   42769,234   DOS ERROR
POKE   42770,234   MESSAGES
POKE   43624,X       SELECTS DISK DRIVE WITHOUT EXECUTING A COMMAND (48K SYSTEM)

POKE   43699,0       TURNS AN EXEC FILE OFF BUT LEAVES IT OPEN UNTIL A FP, CLOSE
POKE   43699,1       TURNS AN EXEC FILE BACK ON.          INIT, OR MAXFILES IS
ISSUE

POKE   44452,24    ALLOWS 20 FILE NAMES (2 EXTRA)
POKE   44605,23    BEFORE CATALOG PAUSE

POKE   44505,234   REVEALS DELETED FILE
POKE   44506,234   NAMES IN CATALG

POKE   44513,67    CATALOG WILL RETURN ONLY LOCKED FILES
POKE   44513,2       RETURN CATALOG TO NORMAL
POKE   44578,234   CANCEL CARRIAGE
POKE   44579,234   RETURNS AFTER CATALOG
POKE   44580,234   FILE NAMES

POKE   44596,234   CANCEL
POKE   44597,234   CATALOG-STOP
POKE   44598,234   WHEN SCREEN IS FULL

POKE   44599,234   STOP CATALOG AT EACH FILE
POKE   44600,234   NAME AND WAIT FOR A KEYPRESS
POKE   46922,96    THIS ALLOWS DISK
POKE   46923,234   INITIALATION
POKE   46924,234   WITHOUT PUTTING
POKE   44723,4       DOS ON THE DISK

POKE   49107,234   PREVENT LANGUAGE
POKE   49108,234   CARD FROM LOADING
POKE   49109,234   DURING RE-BOOT

POKE   49168,0       CLEAR KEYBOARD
POKE   49232,0       DISPLAY GRAPHICS
POKE   49233,0       DISPLAY TEXT
POKE   49234,0       DISPLAY FULL GRAPHICS
POKE   49235,0       DISPLAY TEXT/GRAPHICS
POKE   49236,0       DISPLAY GRAPHICS PAGE 1
POKE   49237,0       DISPLAY GRAPHICS PAGE 2
POKE   49238,0       DISPLAY LORES
POKE   49239,0       DISPLAY HIRES
------------------------------------------------------------------------------
```

48K MEMORY MAP

DECIMAL	HEX	USAGE
0-255	$0-$FF	ZERO-PAGE SYSTEM STORAGE
256-511	$100-$1FF	SYSTEM STACK
512-767	$200-$2FF	KEYBOARD CHARACTER BUFFER
768-975	$300-$3CF	OFTEN AVAILABLE AS FREE SPACE FOR USER PROGRAMS
976-1023	$3D0-3FF	SYSTEM VECTORS
1024-2047	$400-$7FF	TEXT AND LO-RES GRAPHICS PAGE 1
2048-LOMEM	$800-LOMEM	PROGRAM STORAGE
2048-3071	$800-$BFF	TEXT AND LO-RES GRAPHICS PAGE 2 OR FREE SPACE
3072-8191	$C00-$1FFF	FREE SPACE UNLESS RAM APPLESOFT IS IN USE
8192-16383	$2000-$3FFF	HI-RES PAGE 1 OR FREE SPACE
16384-24575	$4000-$5FFF	HI-RES PAGE 2 OR FREE SPACE
24576-38999	$6000-$95FF	FREE SPACE AND STRING STORAGE
38400-49151	$9600-$BFFF	DOS
49152-53247	$C000-$CFFF	I/O HARDWARE (RESERVED)
53248-57343	$D000-$DFFF	APPLESOFT IN LANGUAGE CARD OR ROM
57344-63487	$E000-$F7FF	APPLESOFT OR INTEGER BASIC IN LANGUAGE CARD OR ROM
63488-65535	$F800-$FFFF	SYSTEM MONITOR

PEEK: TO EXAMINE ANY MEMORY LOCATION L, PRINT PEEK (L), WHERE L IS A DECIMAL NUMBER 0-65535. TO PEEK AT A TWO-BYTE NUMBER AT CONSEQUTIVE LOCATIONS L AND L+1, PRINT PEEK (L) + PEEK (L+1) * 256

POKE: TO ASSIGN A VALUE X (0-255) TO LOCATION L; POKE L,X. TO POKE A TWO-BYT NUMBER (NECESSARY IF X>255), POKE L,X-INT(X/256)*256, AND POKE L+1,INT(X/256).

CALL: TO EXECUTE A MACHINE LANGUAGE SUB ROUTINE AT LOCATION L, CALL L.

JUST FOR FUN TRY THIS: POKE 33,90. THEN TRY LISTING YOUR PROGRAM. OR TRY: 0,99 OR POKE 50,250 OR POKE 50,127. USE RESET TO RETURN TO NORMAL.

FOR TRUE RANDOM NUMBER GENERATION TRY THIS:X= RND(PEEK(78)+PEEK(79)*256)

TO LOCATE THE STARTING ADDRESS OF THE LAST BLOADED FILE USE: PEEK(-21902)+PEEK

(-21901)*256 (RESULT IS IN HEX)

**TO DETERMINE THE LENGTH OF THE LAST BLOADED FILE USE: PEEK(-21920)+PEEK(-21919
*256 (RESULT IS IN HEX)**

**TO DETERMINE THE LINE NUMBER THAT CAUSED AN ERROR TO OCCUR, SET X TO: PEEK(218
+PEEK(219)*256**

--

==
DOCUMENT miffins2.txt
==

*** HOW TO USE DEMUFFIN ***

AS THE TITLE SAYS, IT'S CALLED DEMUFFIN PLUS. I'M SURE ALL YOU APPLE USERS KNOW WHAT MUFFIN IS (IT UPDATES DOS 3.2 FILES TO DOS 3.3 FILES. DEMUFFIN PLUS IS A MODIFICATION OF THIS PROGRAM. WHAT IT DOES IS READ THE RESIDENT D.O.S. (THE DOS ON THE DISK YOU WANT TO CRACK) AND 'UPDATES' IT TO STANDARD APPLE DOS DEMUFFIN DOESN'T HAVE THE RESIDENT DOS IN MEMORY, YOU HAVE TO LOAD DEMUFFIN PLUS AT A LOCATION WHERE IT WON'T GET MESSED UP WHEN YOU BOOT UP THE DISK YOU WANT TO CRACK. THIS IS ACCOMPLISHED BY BLOADING IT AT A$4000 OR HIGHER. SINCE THE PROGRAM ONLY RUNS AT LOCATION $803, YOU HAVE TO MOVE IT DOWN IN MEMORY AFTER YOU BREAK INTO THE PROGRAM. HERE IS A SIMPLE DEMONSTRATION ON HOW TO USE DEMUFFIN PLUS:

1) BOOT UP DOS 3.3
2) BLOAD DEMUFFIN PLUS,A$4000
3) BOOT UP DISK YOU WANT TO CRACK
4) HIT RESET WHEN DOS HAS BOOTED UP
5) KEEP HITTING RESET UNTIL DRIVE STOPS
6) CALL-151
7) 803<4000.6000M
8) 803G
9) COPY ALL FILES ONTO 3.3 INIT'ED DISK

NOTE: THIS WILL ONLY WORK ON A DISK THAT WHEN YOU BOOT UP, A CURSOR APPEARS . IT WILL NOT WORK ON SOME OF THESE SOMETIMES. SOME EXAMPLES OF PROGRAMS THAT CAN BE DEMUFFINED WITH NO OTHER CHANGES ARE:
SPACE VIKINGS
KRELL S.A.T. PREPARATION SERIES
CASTLE OF DARKNESS

AND MANY OTHERS!

DEMUFFIN PLUS WILL ALSO NOT WORK ON PROGRAMS THAT HAVE NO DOS (THE ONES THAT JUST LOAD INTO MEMORY).

```
==============================================================================
DOCUMENT ml.part.i
==============================================================================

***************************************
*                                     *
*                                     *
*   MACHINE LANGUAGE TUTORIAL DISK    *
*                                     *
*      WRITTEN BY DR. FIRMWARE        *
*                                     *
*                                     *
***************************************
```

The aim of this disk is for you the
reader to understand machine language
to an extent so that you can program
fully in machine language (ml).

PART I

The fundamentals.

The first part of the course is number
bases. if you undestand binary and
hexadecimal numbers and conversion
between these and decimal, you can skip
to the next section.

Binary: Base two.

Number bases are what we are dealing
with here. The number base that we
normally use in everday life is
decimal. 'Decimal' comes from latin
where it meant ten. We have ten digits,
0,1,2,3,4,5,6,7,8, and 9, which are
combined in various ways to produced
other numbers. It is understood that
the number '345' means 3x100+4x10+5x1.
The right-most digit has the least
significance, while the left-most has
the most significance. From left to
right, the numbers that are multiplied
with the digits are successive powers
of 10. 1=10~0, 10=10~1, 100=10~2, etc.

Now applying these fundamentals, we'll
construct the base two, or binary,
number system. First there are two
digits, 0, and 1. So, the right-most
digit has the least significance and
the left-most, the most significance,
just like in decimal. Now, the numbers
multiplied with the digits will be
successive powers of two. 2~0=1, 2~1=2,
2~2=4, 2~3=8, etc. We now have the

basics down, so we'll take a number,
such as '1001101', and find it's
decimal value.

To start, we'll take the right-most
digit and find out what it is
multiplied with. Since it's the right-
most digit, it's multiplied with two
to the power of zero. 1x2~0=1. Now,
repeat the process, this time with the
second right most digit, which is a 0.
0x2~1=0. Continueing produces: 1x2~2=4,
1x2~3=8, 0x2~4=0, 0x2~5=0, and
1x2~6=64. Summing the results,
1+0+4+8+0+0+64=77. So 77 is the decimal
value of the binary number 1001101.

If you want to practice some, just make
strings of 0's and 1's and do what we
did above.

Conversion from decimal to binary is a
little more complex. Suppose we take a
decimal number, 35. To convert, we do a
series of steps.

1> Divide the number by two, and put
the remainder aside.

2> Replace the dividend with the
quotient.

3> Repeat step 1 & 2 until the number
reaches zero.

4> Take the remainders and place them
in a row, the first is right-most, the
last is left-most.

And that's it. To demostrate, we'll
convert 35 to binary.

```
       0 R=1 -------
     ---          !
   2)  1 R=0      !
     ---          !
   2)  2 R=0      !
     ---          v
   2)  4 R=0      100011
     ---               ~
   2)  8 R=1           !
     ---               !
   2) 17 R=1 ------------
     ---
   2) 35
```

There. Quite simple. The diagram would
look somewhat better on paper, but this

will have to do in the mean while.

Hexadecimal

'Hex', as it is affectionately called
by in most computerese dialects, is
nothing more than a base sixteen number
system. Let's go through some basics.

It has 16 digits. These digits are the
numbers 0-9, and the letters A-F. The
reason why the letters are included is
because there aren't enough numbers.
Let's take a number, $4A. Note that
when you see a '$' infront of a number,
it denotes that the number is a hex
number. $4A means 4x16~1+10x16~0. The
letters are the numbers from 10-15, A
being 10, B is 11, C=12, etc.

Conversion to decimal is exactly the
same as for binary. To demostrate we'll
convert 10234 to hex.

```
            0
       ------
    16)     2  R=7  ----
       ------          v
    16)    39  R=15    7FA
       ------            ~
    16)   639  R=10 -----/
       ------
    16) 10234
```

There we are! 10234 is $7FA.

One interesting fact: since 16=2~4,
then a 4 digit binary number is equal
to 1 hex digit, i.e. 1111=$F, 1010=$A,
etc. This makes binary to hex, and
vice versa, conversion very easy. For
example, the number $3A0 in binary is

```
  0011 1010 0000.
   ~    ~    ~
   !    !    $0
   !    $a
   $3
```

This ends the discussion on number
bases and now the reader should be
aquianted with binary and hex and what
they mean.

Bits and Bytes.

A bit is really only a binary digit.
In other words, a 1 or a 0. These are
digital computers handle, strings upon

strigs of bits. Unfortunately, bits are
very combersome, because even the
charcters that you see require 8 bits
each. The screen size is 40x24, and
that adds up to 7680! bits!

A more convinient form are two digit
hex numbers. A two digit hex number
represents 8 bits in only two digits.
A more common name for this compact
unit is a byte.

You might know that your computer has
64K RAM. The K represents 1,024 bytes.
So this means that your computer has
65,536 bytes of RAM memory. 65,536 can
be expresses more conviniently as 2~16.
This is important for reasons that
we'll discuss a little later.

Well, there we are! Now that we have
some basics down, we can get to some
machine language.

=======================================
DR. FIRMWARE, OCT 21st, 1985.
I CAN BE REACHED ON TESTY, 514-332-6852
OR ON TRANSFERS AE, 514-738-1247
=======================================

```
================================================================================
DOCUMENT ml.part.ii
================================================================================

***************************************
*                                     *
*                                     *
*   MACHINE LANGUAGE TUTORIAL DISK    *
*                                     *
*      WRITTEN BY DR. FIRMWARE        *
*                                     *
*                                     *
***************************************
```

PART II

Machine language command structure.

Even though this sounds complicated, the structure of machine language commands is quite simple. The command is one to three bytes long and consists of two sections, the operator and the arguement. The operator is always one byte long and the arguement is either zero, one or two bytes long. If the arguement is zero bytes long, then it is said that there is no arguement for that command.

The accumulator

The accumulator is the primary register in the 6502 microprocessor. It is an 8 bit register, which means that it can handle only eight bits at a time or the numbers from zero to 255.

To put numbers into the accumulator, we use a command called LDA which stands for LoaD Accumulator. This command takes the value generated by the arguemant and places it into the accumulator.

Addressing modes

Addressing modes are very important. These tell the computer how to deal with the arguement that it recieves. We will only be dealing with two modes for the present, immeadiate, and absolute.

In immeadiate addressing mode, the LDA command load the accummulator with the actual value of the arguement. Suppose that we wanted to load the value $6F into the accumulator. We would do this by telling the microprocessor to 'LDA #$6F'. That is assembly language. In actual fact, the code used by the microprocessor would represent it as '$A9 $6F'. The $A9 tells the microprocessor that you want to load the accumulator in immeadiate addressing mode. The $6F is the arguement and is treated as described above. So then, the number $6F is put directly into the accumulator.

The LDA command in immeadiate addressing mode is two bytes long. The first byte being the operator ($A9) and the second being the arguement.

Memory locations.

The Apple computer has 2~16 memory locations. Each memory location is 8 bits large. Each memory location can be referenced by a 4 digit hex number.
A four digit hex number is 2 bytes long and can be cut in half into two separate bytes. The byte on the left is more significant than the one on the right, so the one on the left is called the Most Significant Byte (MSB) and the one on the right is the Least Significant Byte (LSB).

In absolute addressing mode, the LDA command takes the arguement as an address and then takes the value held in that address and transfers it to the accumulator. The arguement is two bytes long and it forms the address LSB first and MSB second. The address is in effect backwards.

Say you wanted to load the accumulator with whatever was in location $456D. The operator is $AD, this is followed by the LSB which is $6D, and finally the MSB, $45.

Storing the accumulator.

To move the contents of the accumulator to some other memory location, we use the command STA, which stands for STore Accumulator.

The STA command has an absolute addressing mode. The hex operator is $8D and it is followed by the LSB and

MSB, in that order. After the command is executed, the accummulator still contains the value.

Now we can make a tiny program to store the value $8D into location $2000. First, we have to load it into the accumulator. To do this, we'll load the $8D into the accumulator through the LDA immeadiate command. So, then we'll store the accumulator into $2000 while it contains our value using the STA absolute command.

In assembly language, our program looks like this:

```
LDA #$8D
STA $2000
RTS
```

Note: the '#' indicates that the command is in immediate addressing mode. The RTS is going to be used as a general 'end' command for now, until I can explain it's actual usage.

This assembly language version is not understandable by the microprocessor. It has to be translated into hex codes. This translation is normally done by an assenbly program, but since this is a short program, we'll do it by hand.

We are going to put this program at location $300-$306. This area can be used for short programs as $300-$3b0 is free memory space. An extended memory map will be included in a later edition.

```
LDA #$8d   -->    $A9 8D
STA $2000  -->    $8D 00 20
RTS        -->    $60

hex location       contents
---------------------------
 $300               $A9
 $301               $8D
 $302               $8D
 $303               $00
 $304               $20
 $305               $60
```

The program can be entered into memory using the BASIC POKE command. $300 is equal to 768 and the rest of the hex numbers you should be able to convert into decimal yourselves.

This concludes PART II of the series.
Coming next: X and Y registers.

```
=======================================
DR. FIRMWARE, 1985.
I CAN BE REACHED ON TESTY, 514-332-6852
OR ON TRANSFERS AE, 514-738-1247
=======================================
===============================================================================
DOCUMENT ml.part.iii
===============================================================================

***************************************
*                                     *
* PART VII - ASSEMBLERS               *
*       WRITTEN BY DR. FIRMWARE       *
*                                     *
***************************************
```

Assemblers are used for easily writing
up code from mnemonics to hex. To do
this by hand is tedious, to say the
least, and eventually one will make an
error here or there.

Mnemonics are the codes that we have
been using, like 'LDA'. Since these do
not signify the addressing mode, there
is a set of symbols that are normally
used.

To indicate immediate addressing mode,
we put a '#' in front of the arguement.
To indicate absolute addressing mode,
we just put the address. To indicate
indexed absolute mode, we put the base
address followed by a comma and the
indexing register. Here is a short list
of the conventions:

```
LDA #$00           -IMMEDIATE
LDA $0000          -ABSOLUTE
LDA $0000,X        -ABSOLUTE IND. X
LDA $0000,Y        -ABSOLUTE IND. Y
LDA $00            -ZERO PAGE
LDA $00,X          -ZERO PAGE,X
LDA $00,Y          -ZERO PAGE,Y
LDA ($00,X)        -INDIRECT,X
LDA ($00),Y        -INDIRECT,Y
JMP ($0000)        -INDIRECT
INX                -IMPLIED
ASL A              -ACCUMULATOR
```

The modes will be fully explained
further down.

Here'S a simple program in assembly
language:

```
(1)        (2)     (3)        (4)
          ORG     $300       Start at $300
COUT      EQU     $FDED      COUT stands for $FDED
          LDX     #$0C       Load X with length.
LOOP      LDA     TEXT,X     Load A with a chr.
          JSR     COUT       Gosub chr output at $FDED
          DEX               Decrement X by 1.
          CPX     #$00      Is it zero yet?
          BNE     LOOP      If not goto to 'LOOP'
          RTS               Else end.
TEXT      ASC     'DR. FIRMWARE'
                            - ASCII chrs for my name.
```

The columns denoted by the numbers in brackets are as follows: (1) label field, (2) operator field, (3) arguement field, and (4) comment field.

Labels.

Labels are used in assembly language to simplify things. The label 'PLOTIT' means a lot more than $27A5. Note that labels are all one word, no spaces.

In this program, the label 'LOOP' is used to denote a specific place in the program. In the branch statement, 'LOOP' is refered to, and when the program is assembled, the address in memory where 'LOOP' will be is the address the argument the statement will use.

The operator field.

This is where the mnemonics are. The main part of the program is here. However, you might have noticed the 'ORG' and the 'ASC'. These are 'psuedo-ops'. These pseudo-ops tell the assembly program needed information such as the address where the program is supposed to run.

There are many pseudo-ops, and since each assembly program has thier own, it would be hard to cover all of them. So, refer to any manuels that you've copied with your software.

Arguement field.

This field is where the arguements for the operators are, if there need to be any given. The arguments need not to be hex numbers any more. One can use labels for everything, if it pleases you. But in general, since main point of assembly programs is to let the programmer program and not mess around with (yucky) hex numbers, labels in this field seem to be the way to go.

Comment field.

This field is to help narrate your program, that is, to help someone who is reading it (including yourself at times, i'm sure). Of course one can put things like editorials or dirty msgs here, but each to his own.

In this column, i will be using a nice mix of psuedo-ops and comments, so, if this program doesn't work as typed, sue me.

Ok, with that out of the way, here is a description of the previously mentioned addressing modes.

Zero page.

Zero page is somewhat special because the MSB of all the bytes is $00. For this mode, there is only one arguement byte. This byte is the LSB of the address and you will get addresses like $0045.

When indexing zero page with either X or Y, the resulting address is always smaller than $100. For example, LDA $45,X when X holds $FF will read address $44 and put it in the accumulator. The logic goes thus: $45+ $FF= $144. Because the result is greater than $100, the one at the front is dropped and all you have left is $44.

JMP.

This is a goto-like command in m.l. and can be considered as such. The command has 2 argument bytes and these represent the address where program execution will continue in the form LSB MSB. Note the address to jump to is backwards just like the LDA command in absolute mode.

Indirect jump.

The indirect jump is variation on the JMP, such that the argument forms an address from where the actual 'jump to' address is found. (Both in MSB LSB form.)

Suppose there was such an incident:

```
300: JMP ($800)
.
.
800: $00 $20
```

($800 Contains $00 and $801 contains $20)

From $300, the argument gives $800. The program goes and gets $800 and $801 and re-arranges them to give $2000. Then the program jumps to $2000 and continues execution.

A very useful command at times.

Well, unfortunately the indirect commands will have to wait 'til next time.

```
***************************************
* DR. FIRMWARE CAN BE REACHED ON THESE*
* BOARDS: 514-738-6576  TRANSFERS     *
*         514-744-4108  APPLE ENCH.   *
***************************************
```

```
================================================================================
DOCUMENT ml.part.iv
================================================================================

***************************************
*   MACHINE LANGUAGE TUTORIAL PART IV *
***************************************
```

The CMP command

CMP stands for CoMPare accumulator. It has an immediate mode. In immediate mode, what happens is this: the value of the arguement is subtracted from the contents of the accumulator and this result is dicarded except for the effects on the zero, negative and carry flags. English translation immediately following.

We'll take it slowly. The value of the arguement is the byte following the operator. This value is subtracted from the contents of the accumulator. Say the arguement is $40 and the accumulator holds $60. $40 is subtracted from $60 and you get $20. Now supposing the arguement value is greater than the value in the accumulator, that is $60 is subtracted from $40. Doing this algebraicly, you would get -$20, but the accumulator can only hold numbers from zero to 255. So what happens is that the microprocessor adds $100 to it. -$20+$100=$D0. And from this number, the flags take thier cues. By the way, this resulting number is thrown out and forgotten about.

What are flags??

The flags that were mentioned live in the status register. Also called the 'p' register. This register is an 8-bit register and a flag is one of these bits. However, even though there are eight bits in the register, there are only seven flags. These flags are:
1> Carry
2> Zero
3> Interupt disable
4> Decimal mode
5> Break command
6> Overflow
7> Negative.

For now, we will deal only with the

zero, and negative flags. The zero flag is set to 1 whenever a zero is loaded, stored, or gotten as a result in an arithmatic command, such as the CMP. The negative flag is set to one whenever a negative number is loaded, stored, or gotten as a result in an arithmatic command. This machine of ours defines a negative number as any number that has its highest bit set to 1. That is, any number greater than $7F.

Branching.

In essence, what the branching statements do is this. They check a specific flag and then depending on whether that flag is a 1 or a zero, go to a location specified by the arguement. For example, BEQ branches when the zero flag is set to 1. (If the flag is a 0, program flow continues on with the statement following the branch command.

Let's take a look at a short little program.

```
300:A2 00        LDX #$0
302:BD 11 03     LDA $0311,X
305:C9 00        CMP #$0
307:F0 07        BEQ OUT
309:20 ED FD     JSR $FDED
30C:E8           INX
30D:4C 02 03     JMP $0302
310:60       OUT RTS
311:C4 D2 AE ...  (This is hex
representation of text and ends with a
$0)
```

This program will print out whatever the text says (text is at $311 and is in ASCII chrs) using a ROM routine at $FDED which prints characters onto the screen. The text must end with a $0 and be less than 255 bytes long, otherwise you will either hang the system or fuck it up royally.

Even though there are quite a few new commands in the program, we will only focus on the role of the BEQ command.

The second byte of the command (BEQ is a 2-byte command) is the arguement and determines where the program branches to, if it branches.

The way which the address of the branch
is determined is this. The M.P.U takes
the address of the next command after
the branch statement (in our program
above, this address would be $309) and
adds the value of the arguement to it.
In our program, the arguement for the
branch statement is $7. $7+$309=$310.
Which is what we want to happen. But,
all number from $80 up are negative!
So if the arguement had been $F8, $F8
is equal to -$8 and $309-$8=$301. The
program would have jumped to $301.

BNE

BNE is a branch command that branches
when the Z flag=0, that is a non-zero
number is stored, loaded or gotten in
an arithmetic operation. This command
works in exactly the same way as the
BEQ to generate the 'branch-to'
addresses.

A word on the other commands.

In the program, there were a lot of new
commands used, i will cover them in the
near future, but just to give you an
idea, here are some quick defenitions.

JMP: this is much like a BASIC 'goto'.

JSR: this is much like a BASIC 'gosub'.

INX: increment X by 1. (add 1 to the value in the X register)

RTS: this is a general 'end-of program'
statement. there are some better uses
which we will cover.

Well, that's all folks!

```
***************************************
*                                     *
*       Dr. Firmware's M.L            *
*            tutorial                 *
***************************************
*                                     *
* TESTAMENT:(514)-332-6852            *
* TRANSFERS AE:(514)-738-1247         *
*                                     *
***************************************
```

==
DOCUMENT ml.part.v
==

```
***************************************
*                                     *
*  M.L. PART FIVE   BY DR. FIRMWARE   *
*                                     *
***************************************
```

This part is going to be about the arithmetic and logic unit of the 6502. The ALU is what does the addition and subtraction and bit operations. Presently, we will only cover the math, leaving the bit operations for later.

If you read the previous column, you will have noticed that the CMP 'subtracts' two numbers. this subraction takes place in the ALU.

To subtract two numbers, we use the SBC command. In immediate addressing mode, the arguement is subtracted from the value currently held in accumulator, and the result is then put back into the accumulator. It is a fairly simple procdure, but this is not all there is to it. First of all, negative numbers are represented as $100+the number. Also, there is the carry flag to deal with. This flag was put into the formula so that calculations involving numbers greater than 255 (that is 1 byte) could be simplified. Once the result of the A - arguement is fonud, then the oppisite (techniquely called the two's complement, see below) of the carry (that is, if C=1, then use 0 and vice versa) is subtracted from the result. (The carry is a one bit flag and can only hold 0 or 1, so if it is set the wrong way, the answer will be off by one.) Since we want to use this command to produce right results, we must set C=1. This is done by an SEC. To subtract 2 numbers, the following routine should be used.

```
SEC
LDA #FIRST NUMBER
SBC #SECOND NUMBER
```

The SBC command also has absolute, indexed X and Y modes.

Adding.

Adding numbers is very similar to subtraction. The ADC (add with carry not analog to digital converter) command adds the value in the accumulator to the arguement plus what the carry flag is set to. To set the carry to 0, we use CLC. Here's the routine:

```
CLC
LDA #FIRST NUMBER
ADC #SECOND NUMBER
```

And your desired result is in the accumulator. As we said earlier, the way the carry functions allows us to add mutli-byte numbers easily. Suppose we have two 3-byte long numbers. We will represent these numbers by the following method. N1 will be used to denote the first number and N2 the second. B1 will be used to denote the left-most (MSB) of each number and B2 B3 as the successive bytes. So the numbers are N1B1.N1B2.N1B3, and N2B1.N2B2.N2B3. We will add the LSB's first and then follow with the middle bytes and finally the MSB's. For the LSB's, we will set the carry to zero. This will give us the answer we want for the LSB of the result (RB3). After storing RB3 in it's proper place, we will then add N1B2 and N2B2 together, leaving the carry as it is. After an addition is made, the carry is set to 0 if the result is less than 255, and set to 1 if it is greater than 255. The result can range from 0 to 510, which can be represented in 9 bits, C+ the accumultor. Now if the result is greater than 255 for the LSB's, we want to add one to the next result of the middle bytes. This is automaticly done by the carry. So, here is the routine:

```
CLC
LDA #N1B3
ADC #N2B3
STA (THE ADDRESS OF) RB3
LDA #N1B2
ADC #N2B2
STA (THE ADDRESS OF) RB2
LDA #N1B1
ADC #N2B1
STA (THE ADDRESS OF) RB1
RTS
```

The result is C.RB1.RB2.RB3. The reason why the carry is at the top is because if you add $FFFFFF and $FFFFFF you get $1FFFFFE. The one is the carry. It is advisable to set up 'registers' in RAM so that a generalized addition routine can be utilized. What it means is that you've set aside nine byte (say $300 to $308) to be three 3-byte registers. One from $300-$302, which would be where N1 would be stored, another from $303-$305 , resting place for N2, and the last from $306-$308, for the result (R). You would have to figure out something with the carry though. To help you with this there are two branch commands BCC and BCS which branch on carry clear (C=0) and carry set (C=1), respectively.

Another possibility is to make an indexed addition routine using the X register as a counter. Though I won't give the code here, by examining the code given in the previous column and the addition routine, it can be worked out quite simply.

To subtract multi-byte numbers, we can use the same routine as above, except replacing the CLC with a SEC and the ADC's with SBC's. This works, though the result would now be RB1.RB2.RB3 with the carry telling you whether the result is negative or positive. If C=1 then the result is positive and vice versa. However, if the result is negative, the number is represented as $1000000+result.

Next time round: assemblers, monitor, and other fun stuff.

```
***************************************
*                                     *
* CALL THESE BOARDS:                  *
*    TESTAMENT: (514)-332-6852        *
*    GAMMA-LINE: (514)-683-9176       *
*    TRANSFERS II AE/CAT: 738-1247    *
*                                     *
***************************************
```

Oh yeah, since you asked, 2's complement is gotten by taking the next highest power of 2, and subtracting one form it. Then, subtract your number from that result and voila. For example the next highest power of 2 after 1, is

2. Minus one is one and then 1-1=0.

The negative numbers sort of work on
the same principle, except, the one is
not subtracted and it is the 256's
complment.

So long for now..

```
================================================================================
DOCUMENT ml.part.vi
================================================================================

***************************************
*                                     *
* PART VI OF DR. FIRMWARE'S M.L. TUT. *
*                                     *
***************************************
```

As was previously said, this article is about monitor, assemblers and other methods of entering M.L. programs into memory.

Poking and calling.

To enter a program into memory from BASIC, one can POKE the decimal equivalents of the hex op-codes (the values that the microprocessor understands) into the appropriate range of memory and then calling the subroutine with a 'CALL' statement. This method is quite tedious and complicated due to the fact that one would have to derive the hex codes by oneself by looking them up in the Apple reference manual supplied. This may prove to be even more difficult if one has no such manual.

Monitor.

Monitor is located in the range of memory from $F800-$FFFF. To get into monitor, type 'CALL -151' from the BASIC prompt. A '*' should appear with the cursor beside it. Now you are in monitor. There is a different set of commands availible to you than in BASIC The most simple of these is the <CR> (or carriage return (ctrl-m)). This will display the next 8 location and thier values. to look at a particular location, just type the hex equivalent of the location (ie $300, except with-out the '$' in front). Pressing return will then give you the next 8 locations and their values.

To change the value of a specific location, we must type the location, (in hex, with out the '$', as above) a ':' and then the value we want to change it to. For example, suppose we wanted to change the value in location

$300 to a $A9, we would type the
following:

```
*300:A9
~
```

'The '*' is the prompt, so don't type
it, it is included here (and most else-
where) as a convention. (Oh yeah, add a
<CR> to the end.)

In BASIC, to do this, we would have to
'POKE 768,169'. Note that the '300' is
the hex equivalent of '768'

Ok, but suppose we wanted to change a
whole bunch of locations in a row, and
not just one. There is an easier way
than to type each location, a colon and
then the value. you can just type the
first location, then follow with as
much data as you can (in hex) spacing
between each data element. Like this:

```
*300:A9 C1 20 FD ED 60
```

This puts $A9 in $300, $C1 in $301, $20
in $302, etc. There is, of course, only
254 characters that you can enter at
one time, but it does cut down on the
typing. There is another good feature
of monitor that one can make use of
which allows you to continue entering
values from the point you left off at.
To use this, after entering the first
bunch of numbers, you can just type a
colon and then whatever data, and it
will automatically put into the next
location. Like this:

```
*300:A9 C1 20
*:FD ED 60
```

This will have exactly the same as
result as the previous example. Note:
When entering data using this feature,
it is wise that if you get distracted
and go elsewhere to fiddle for awhile,
you should then type the location of
the next location, otherwise it may be
put your data somewhere where it is not
appreciated.

Ok, so you've typed your program in.
Now you want to check it if it was
entered properly. You can always just
use the <CR> command and check, but
there is a somewaht easier way. The
Monitor has a feature which does

partial disassemblies. To use it, type
the location and then an 'L'.
Like this:

```
*300l
```

What you should see on this screen (if
you have done the steps previously
outlined, will look something like
this:

```
0300-A9 C1      LDA #$C1
0302-20 ED FD   JSR $FDED
0305-60         RTS
0306-00         BRK
0307-00         BRK
etc...
```

This first column (before the '-') is
what address (location) we are looking
at. The second column is the hex codes
contained in the addresses. The third
column hold the mnemonics (more on this
later.)

You will notice that the addresses do
not increment by ones, but by the
number of numbers on the right of them.
This is because the commands are not
all the same length, but vary according
thier addressing modes (we've done
immediate, absolute, and indexed, but
more on these later)

Anyway, these are the basic commands
used in monitor. There are commands for
moving and comparing ranges of memory.
Thier syntax is as follows.

To move a range:

```
*(DEST)<(START).(END)M
```

For example:
```
*9600<C600.C700M
```
will move the memory in C600 to C700 to
9600. That is, the value in $C600 will
be stored in $9600, the value in $C601
will be stored in $9600, etc. This
command's use may not be clear to you
but, it is quite helpful at times.

Anyway, so long for now, since i ran
out of space, i will do assemblers in
the next segment.

```
****************************************
*                                      *
* DR. F CAN BE REACHED ON THESE BBS'S *
*   TESTAMENT: (514)-3326852          *
*   TRANSFERS (AE/CAT):514-7381247    *
*                                     *
***************************************
```

```
==============================================================================
DOCUMENT oneguy.txt
==============================================================================

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
 Hi guys, this is The Wyvern here, and i just wanted to share this little piece
of info i grabbed out of Softline some time ago. Its pretty funny at that, and
its about this one guy who pirated a copy of Zork, but did a bad thing by
mistake, not knowing the results...Well enjoy, and remember to ring up The
Temple of Doom at [805] 682-5148 as soon as you can!
================================================================================
(ring)
"Hello, i have a problem with Zork."
"What kind of problem?"
"Well, i went to save it to disk."
"Yeah?"
"and it saved."
"So?"
"Oh. I left my program disk in the drive."
"And?"
"And so now when i run the program, it just spits out call statements."
"Well, on our copy of Zork, it tells you to take out the program disk and put
 the save disk in before you save."
"Oh."
"Do you have an original copy?"
"A Copy."
"You should have gotten an original."
"Oh."
"Bye."
"Bye."
(click)

Well there you have it, next it says 'best tip you will get all week', and they
mean to say that, to buy the original and not pirate others. But thats not the
wayyou should take it, still pirate them but take this tip and dont save them to
the same disk. Oh well, something to read, no?
================================================================================
===============================================================================
DOCUMENT oo.world.info
===============================================================================
```

```
[ Subject ] :Out of This World GS (Demo?)
[ From    ] :The Magnet (#1)
[ To      ] :All
[ Date    ] :05/21/92  03:01:45 PM
```

I'm not sure if Out of This World is a demo or not, but I can say it's probably not the 100% complete version. BUT I have reason to believe that his version is about 95% complete.

When you first run it, it asks you for your viewing preference:
  16mm means the entire screen ; this mode is sluggish even on my 7Mhz Zip GS
  34mm is about 2/3 of the screen ; it's bearable and is recommended by me
  70mm is only half the screen, very narrow sorta like watching a movie on TV
  Television is same as 70mm from what I've seen.. I can't tell any difference.

For the controls, I can't get joystick to work anyways, so lets just say keyboard. or either one you pick don't matter since you can use keyboard.

It seems that you just watch the guy drive up in the Ferrari, do his experiment and it fails and he gets blown into Another World, or blown Out of This World (whichever name you choose, since in IBM, both names have been used and released, same game). After this it seems to have just freeze. This is the FIRST and ONLY program ever I've seen on the GS that disables not only control-reset, and not only control-oa-reset, but even the fake cold boot of control-oa-OPTION-reset!! The only way is to turn off the computer!! Holy cow to Bill..

Anyways, upon my further examination, I found out the following keys:

A - Accept/Action (acts like return when choosing PW)
Q - Quit the game/demo
C - Choose Password to different levels (use arrows to move cursor)

OK: There are 22 level codes to this game and I've tried all of them and they all work except for the last one. This is how to do it:

After the demo, or while in the demo, press C anytime and password screen will come up. Enter the 4 letter PW in my pw list file and you will go to that level! While in the game:

A - Shoot (if you have the laser gun) or Step On worms and leeches (when first
    starting game.
Z - Jump
Arrows - move in that direction. Hold down A and Arrows to Run in direction.

After you pick the level, you're actually playing the game!! This is a very difficult game from what I've played, especially since I haven't figured out how to use the joystick.

When shooting with A : hold down for a few seconds to build a shield in front of you that'll block off the enemy's laser. You have to build shield, hide behind shield and try to shoot the enemy. It might sound easy, but remember that your enemy is doing the same and he's much better at it than you are.

Anyways, that's about all I've found out about this game, and I'm including this file in the Out of This World archieve so it can help people get started...actually let me tell you for the first level after the introduction: You and your desk appear under water, you have to hold down the UP arrow to swim up right away or you'll be pulled down under and die. Good luck!

The Magnet

==
DOCUMENT opcodez.app
==

OK,
This is Rich again. I found those Op. Codes for you. If you need any of the other material on them, they are in the 18th issue of Hardcore Computist. Other than this list of Op. Codes, all there is on the 65c02 is a Question/Answer section.

So here's the list:

New Instructions

Instructions	Op Code	Description
BRA	80	Branch Alway (Realitive)
PHX	DA	Push X on Stack
PHY	5A	Push Y on Stack
PLX	FA	Pull X from stack
PLY	7A	Pull Y from Stack
STZ	9C	Store a zero in memory (Absolute)
STZ	9E	Store a zero in memory (Absolute,X)
STZ	64	Store a zero in memory (Zero page)
STZ	74	Store a zero in memory (Zero page,X)
TRB	1C	Test + reset bits w/Accum. (Absolute)
TRB	14	Test + reset bits w/accum. (0 Page)
TSB	0C	Test + set bits w/accum. (Absolute)
TSB	04	Test + set bits w/accum. (0 Page)

Additional Instruction Addressing Modes

Instruction	Op Code	Description
ADC	72	Add memory to Accum. w/carry (Zero Page
AND	32	AND memory with Accum. (Zero Page)
BIT	3C	Test mem. bits w/Accum.(Zero Page,X)
BIT	34	" " " " " " " "
CMP	D2	Compare mem. w/Accum.(Zero Page,X)
EOR	52	Exclusive OR w/Accum.(Zero Page)
JMP	7C	Jump (Absolute(Indirect,X))
LDA	B2	Load Accum. W/memory (Zero page indir.)
ORA	12	OR memory w/Accum. (Zero Page)
SBC	F2	Subtract from Accum. w/borrow (0 page)
STA	92	Store Accum. in memory (Zero page)

That's the best I can do. There were no further articles on the

Instructions. I hope it helps you out. Catch you later.
  Some Dude (Rich)

```
=============================================================================
DOCUMENT param2.app
=============================================================================

Nibbles Away ][ Parameters, Courtesey of The Captain's Log   [612] 377-7747

COMPANY NAME:                                      AUTO-LOAD FILE
PROGRAM NAME          COPY TRACKS    PARAMETERS TO CHANGE        TO USE
----------------    -----------    --------------------   ----------------

A D V E N T U R E   I N T E R N A T I O N A L:
Eliminator -------- 0-21...........ADDR=D5 AA 96
                               SECTMOD [F=16, C=OFF, T=03, S=0D]
                                  CHANGE ADDRESS 2E FROM 20 TO EA
                                  CHANGE ADDRESS 2F FROM 30 TO EA
                                  CHANGE ADDRESS 30 FROM 72 TO EA

A P P L E   C O M P U T E R:
Visicalc /// ------ 0-22...........SYNC
Apple Writer /// -- 0-22...........SYNC
Apple Logo -------- 0-22...........ADDR D5 AA 96
                1-1............ADDR AA D6 EE
                             NIBBLE COUNT=Y
                             FIND MAX = 03
                             SHIFT N+ = 08
                             SHIFT N- = 00

Apple Writer ][ --- 0-3............ADDR D5 AA DA (OR D5 AA DB)
                4-22...........ADDR D5 AA 96
Super Pilot ------- 0-0............ADDR=D5 AA 96
                2-22  SECTMOD [F=16, C=OFF, T=0, S=0A]
                       CHANGE ADDRESS 79 FROM 43 TO EA
                       CHANGE ADDRESS 7A FROM 41 TO EA
                       CHANGE ADDRESS 7B FROM C6 TO EA
Elementary, my
Dear Apple ---------0-22...........ADDR=D5 AA 96

A U T O M A T E D   S I M U L A T I O N S:
Temple of Apshai -- 0-22...........ADDR=D5 AA B5
Crush, Crumble,
& Chomp------------ 0-2  SYNC.......ADDR=D5 AA DB
       ------------ 3-22 SYNC.......ADDR=D5 AA 96

A V A N T E - G A R D E:
Hi-Res Secrets ---- 0-22...........ADDR=D5 AA 96
Zero Gravity Pinball 0-22...........ADDR=D5 AA B5

B P I:       (REVISED)
Accounting -------- 0-22...........ADDR=D5 AA 96
System                              FIX AMNT=04,  GAPBYTE1=C8
                              GLOBAL MOD BYTE D972 FROM 03 TO 00
                11-11..........INS=AD FB E6 FF E6
                           SYNC SIZ=0A

B R O D E R B U N D   S O F T W A R E:
Apple Panic ------- 0-D
Genetic Drift ----- 0-0............ADDR=D5 AA B5
```

```
                1-3............ADDR=BB D5 BB
                4.5-6 BY 1.5
                7.5-B.5
                D-D............ADDR=D4 D5 BB
                E.5-12.5.......ADDR=AD B5 DE

Space Quarks ------ 0-0............ADDR=D5 AA B5
                1-2............ADDR=FF DF DE, DATA MAX=25
                3.5-5.5
                7-9 BY 2
                A.5-B.5
                D-15

Space Warrior ----- 0-0............ADDR=D5 AA B5, DATA MAX=30
                2.5-3.5........ADDR=DF AD DE
                5-8 BY 3
                6.5-6.5
                A-10 BY 3
Warlords ---------- 0-F............ADDR=D5 AA B5

B U D G C O:
Raster Blaster ---- 0-0............ADDR=D5 AA 96, SYNC
                           DATA MIN=18, DATA MAX=40
                5-11 BY 4......ADDR=AD DE, DATA MIN=13, SYNC
                6-12 BY 4......SYNC
                7.5-F.5 BY 4...SYNC
                1.5-3.5 BY 2...SYNC

C A V A L I E R   C O M P U T E R:
Microwave --------- 0-22...........ADDR=D5 AA 96
                            SECTMOD [F=16, C=ON, T=02, S=01]
                              CHANGE ADDRESS DA FROM A9 TO AD
                              CHANGE ADDRESS DB FROM 60 TO 03
                              CHANGE ADDRESS DC FROM 8D TO 81
                              CHANGE ADDRESS DD FROM 7E TO 60

C E N T R A L    P O I N T   S O F T W A R E:
Copy ][ Plus ------ 0-F.............NORMAL (DEL BYTE =20)

C O N T I N E N T A L   S O F T W A R E:
Guardian ---------- 0-1............ADDR=D5 AA B5
                2-11...........ADDR=D6 AA B5
                              INS=DF AA EB F7, SYNC SIZ=0A

D A T A   M O S T:

County Fair ------- 0-22...........ADDR=D5 AA B5
Snack Attack ------ 0-22...........ADDR=D5 AA B5
(REVISED)                    SECTMOD [F=13, C=OFF, S=01, T=00]
                              CHANGE ADDRESS 39 FROM 38 TO 18

Swashbuckler ------ 0-22...........ADDR=D5 AA 96
Casino 21                     SECTMOD [F=16, C=OFF, S=03, T=00]
                               CHANGE ADDRESS 42 FROM 38 TO 18

Space Kadet ------- 0-22...........ADDR=D5 AA 96
Mars Cars                        OVERIDE STANDARDIZER = Y
Crazy Mazey
```

```
Tax Beater -------- 0-22...........ADDR=D5 AA 96
Reap                              SECTMOD [F=16, C=OFF, T=0, S=03]
                                     CHANGE ADDRESS 42 FROM 38 TO 18

Money Muncher ----- 0-22...........ADDR=D5 AA 9
6

D A T A   S O F T:
Dung Beetles ------ 0-0............ADDR=D5 AA B5
                1-1............ADDR=F5 F6 F7
                4-22
                            SECTMOD [F=13, C=ON, T=00, S=01]
                              CHANGE ADDRESS 6D FROM 01 TO 7B
                              CHANGE ADDRESS 6E FROM 61 TO 69

D O N' T   A S K   SOFTWARE:
Word Race --------- 0-1B SYNC......ADDR=D5 AA 96
Claim to Fame &
Sports Derby ------ 0-1B SYNC......ADDR=D5 AA 96

E D U W A R E:
The Prisoner ------ 0-22...........SYNC
Algebra I --------- 0-22...........ADDR=D5 AA B5
Empire 1 World ---- 0-22...........ADDR=D5 AA 96
Builders        3-3............NIBBLE COUNT
Prisoner ][ ------- 0-22...........ADDR=D5 AA 96
                              SECTMOD [F=16, C=ON, T=1F, S=0E]
                               CHANGE ADDRESS D5 FROM AD TO 2F
                               CHANGE ADDRESS D6 FROM 99 TO AF
                               CHANGE ADDRESS D7 FROM F0 TO 32

G E B E L L I    S O F T W A R E:
Firebird ---------- 0-0............ADDR=DD AD DA, SYNC
                1.5-B.5........SYNC

H I G H L A N D   C O M P U T E R   S E R V I C E S:
Creature Venture --- 0-2...........ADDR=D5 AA B5

H O W A R D S O F T:
Tax Preparer ------ 0-22...........ADDR=D5 AA 96

I N F O C O M:
Deadline ---------- 0-22...........ADDR=D5 AA 96
Starcross --------- 0-22...........ADDR=D5 AA 96

I N N O V A T I V E   D E S I G N   S O F T W A R E:
Pool 1.5 ---------- 0-15...........ADDR=D5 AA B5
                1E-21
                        SECTMOD [F=13, C=OFF, T=0B, S=07]
                         CHANGE ADDRESS 6A FROM 8D TO 60

I N S O F T:
Electric Duet ----- 0-22...........ADDR=D5 AA 96
                                    INS= DE AA EB
                                    OVERIDE STANDARDIZER = Y
                                    FIX AMNT=04
```

```
I N T ' L   S O F T W A R E   M K T G:
Math Magic -------- 0-22...........NORMAL

I D S:
Prism Print ------- 0-21...........ADDR=D5 AA 96
                              OVERIDE STANDARDIZER = Y
                              SECTMOD [F=16,C=ON,T=21,S=00]
                               CHANGE ADDRESS 27 FROM FB TO 22

L E A R N I N G   C O M P A N Y:
Bumble Games ------ 0-22...........ADDR=D5 AA 96
Bumble Plot                    NOTE: WRITE PROTECT BEFORE BOOTING!
Rocky's Boots
Juggler's Rainbow

L J K E N T E R P R I S E S:
Letter Perfect ---- 0-22...........ADDR=D5 AA B5

L E V E L   1 0   S O F T W A R E:
Neutrons ---------- 0-22...........ADDR=D5 AA 96
Kaves of Karkhan

L I G N T N I N G   S O F T W A R E:
Master Type ------- 0-2............ADDR=D5 AA B5
                3-22...........ADDR=D4 AA B5
                           (Error on $1B OK)
                SESiz=0A
                1-22...........ADDR=D4 AA 96

M I C R O F U N:
Miner 2049er ------ 1-22...........ADDR=D3 96 F2
                0-0............ADDR=D5 AA 96   NIBBLE COUNT = Y

M I C R O L A B:
Jigsaw ------------ 0-0............NORMAL
                A-17...........NORMAL
                1-9............ADDR=D3 96 F2

M U S E:
Best of Muse ------ 0-22...........SYNC
Three Mile Island
Global War

M I C R O S O F T:
Olympic Decathalon  0-22...........ADDR=D5 AA B5

O N L I N E   S Y S T E M S:
General Manager --- 0-22...........ADDR=D5 AA 96  V1.5
                               SECTMOD [F=16, C=ON, T=1F, S=0E]
                               CHANGE ADDRESS C1 FROM -- TO 4B
                               CHANGE ADDRESS C2 FROM -- TO E0
                               CHANGE ADDRESS C3 FROM -- TO 49
                               SECTMOD [F=16, C=ON, T=21, S=01]
                               CHANGE ADDRESS 2E FROM -- TO 60
Sabotage ---------- 0-22...........NORMAL
Alien Rain
Snoggle ----------- 0-22...........ADDR=D5 AA B5
```

```
Time Zone v1.1 ---- 0-22...........ADDR=D5 AA 96
                               SECTMOD [F=16, C=ON, T=03, S=0B]
                               CHANGE ADDRESS F0 FROM 20 TO EA
                               CHANGE ADDRESS F1 FROM 00 TO EA
                               CHANGE ADDRESS F2 FROM 17 TO EA

Dark Crystal  ------ 0-2 SYNC.......ADDR=D5 AA 96  (OVERRIDE STANDARDIZER = Y)
 DISK 1, SIDE A ---- 3-22...........ADDR=D5 AA 96  (OVERRIDE STANDARDIZER = Y)

Ultima II ---------- 0-22...........ADDR=D5 AA 96

P E N G U I N    S O F T W A R E:
Pie Man ----------- 0-22...........ADDR=D5 AA 96

Transylvania--------0-22 BY 2......ADDR=D5 AA 96 (OVERRIDE STANDARDIZER = Y)
          --------1-21 BY 2......ADDR=D4 AA 96 (OVERRIDE STANDARDIZER = Y)

P H O E N I X    S O F T W A R E:
Zoom Graphics ----- 0-22 BY 2......ADDR=D5 AA 96
2nd edition                          INS
=DD AA ED B5
                1-21 BY 2......ADDR=D4 AA 96
                            N O T E: WRITE PROTECT BEFORE BOOTING!!

Adventure in Time - 0-C...........NORMAL
Birth of the ------ 0-9...........NORMAL
Phoenix

P I C A D I L L Y  S O F T W A R E:
Falcons ----------- 0-0............ADDR=D5 AA B5
                1.5-4.5 X 1.5....ADDR DF AD DE
                5.5-5.5 X 1
                7-A X 1
                B.5-E.5 X 1.5
                10-12 X 1
                13.5-14.5 X 1
                16-19 X 1.5
                1A-1B.5 X 1.5

P R O F E S S I O N A L  S O F T W A R E  T E C H N O L O G Y:
Executive --------- 0-22...........ADDR=D5 AA 96, OVERRIDE STANDARDIZER=Y
 Briefing System                SECTMOD [F=16, C=ON, T=21, S=00]
                                   CHANGE ADDRESS 27 FROM FB TO 22

Q U A L I T Y    S O F T W A R E:
Bag of Tricks -----0-0.............ADDR=D5 AA 96
            -----1-15............ADDR=D6 AA B5
                                SECTMOD [F=13, C=OFF, T-0, S=8,]
                                   CHANGE ADDRESS A0 FROM 20 TO 60

R I V E R B A N K  S O F T W A R E:
International ----- 0-C............ADDR=FF FF FF AA
Grand Prix

S E N S I B L E  S O F T W A R E:
Image Printer ----- 0-2............ADDR=D5 AA 96
                3-7............ADDR=F7 AA 96
                9-22
```

```
                         SECTMOD [F=16, C=OFF, T=0, S=03]
                            CHANGE ADDRESS 42 FROM 38 TO 18
                         SECTMOD [F=16, C=OFF, T=2, S=03]
                            CHANGE ADDRESS 2A FROM 2C TO 4C
                            CHANGE ADDRESS 2B FROM 06 TO 5D
                            CHANGE ADDRESS 2C FROM B7 TO B4
Super Disk Copy --- 0-22...........ADDR=D5 AA 96
                   (VERSION 3.7)          ERRORS OK
The Bug ----------- 0-0............NORMAL
                15-15..........GAP BYTE 2=FF
                           GAP SIZE=10
                16.5-16.5

S E N T I E N T   S O F T W A R E:
Gold Rush --------- 0-22...........ADDR=D5 AA 96

S I L I C O N    V A L L E Y   S O F T W A R E:
Word Handler II --- 0-0............ADDR=D5 AA 96
                11-22
                1-C............ADDR=FF DF DE

S I R I U S   S O F T W A R E:

Autobahn ---------- 0-0............SYNC
                4-6............SYNC
                9.5-C.5........SYNC

Beer Run, Epoch --- 0-0............ADDR=DD AD DA, DATA MAX=25, SYNC
Copts & Robbers,    1.5-13.5.......SYNC
Hadron, Snake Byte
NOTE: Errors  will begin to occur somewhere between track C.5 and track 13.5,
      depending on the particular disk.  This is normal.

Gorgon ------------ 0-0............ADDR=DD AD DA, DATA MAX=25, SYNC
                1.5-C.5........SYNC
                E.5-E.5........SYNC
                D.5-D.5........ADDR=D5 AA B5, SYNC

Sneakers ---------- 0-0............ADDR=DD AD DA, SYNC
                1.5-C.5........SYNC
                D.5-D.5........ADDR=D5 AA B5, SYNC

Gamma Goblins ----- 0-0............ADDR=DD AD DA, SYNC
                1.5-B.5........SYNC
                D-D............ADDR=FF FF FF D5 AA EE
                           DATA MAX=30

Orbitron ---------- 0-0............ADDR=DD AD DA, DATA MAX=25, SYNC
                1.5-E.5........SYNC
                F.5-F.5........ADDR=FF B5 D5 AA

Kabul Spy --------- 0-21...........ADDR=D5 AA 96
(BOTH SIDES)                            SECTMOD [F=16, C=OFF, T=0, S=0]
                              CHANGE ADDRESS 49 FROM -- TO EA
                              CHANGE ADDRESS 4A FROM -- TO EA
                              CHANGE ADDRESS 4B FROM -- TO EA

Outpost ----------- 0-0............ADDR=DD AD DA, SYNC
```

```
                1.5-9.5........SYNC
                B.5-B.5........ADDR=D5 AA AD, DATA MAX=25

Pulsar ][ --------- 0-C
                13-19
                1A.5-1D.5

Dark Forest ------- 0-0............ADDR=DD AD DA, SYNC
                1-22...........ADDR=D5 AA A5, SYNC
                   (Errors on 6-8 and last few tracks OK)
Dark Forest ------- 0-22..........ADDR=D5 AA B5
(#2)                           OVERIDE GLITCH DETECT

Twerps ------------ 0-0............ADDR=DD AD DA, SYNC
                1.5-E.5........SYNC
                1A-1A

Borg -------------- 0-0............ADDR=DD AD DA, SYNC
                1.5-B.5........SYNC
                D-20...........SYNC

Wayout ------------ 0-1C...........ADDR=AD DA DD
                22-22..........ADDR=AA D5 D5 FF D6 FF FD
                21-21..........ADDR=AA,  USE NIBBLE COUNT
                                    SYNC SIZ=0A, MATCH NM=06

S I L I C O N    V A L L E Y   S O F T W A R E:
Word Handler ][ --- 0-0C...........
ADDR=FF DF DE
                11-22..........ADDR=D5 AA 96

S O F T A P E:
Draw Poker -------- 0-22..........ADDR=D5 AA B5

S O F T W A R E   P U B L I S H I N G    C O R P.:
PFS/PFS Report ---- 0-0............ADDR=93 F3 FC FF
                                    INS=93 F3 FC FF
                                    OFFSET -2, SYNC SIZ=0A
                1-13...........ADDR=D5 AA 96,   INS=D5 AA 96
      NOTE: Write Protect the backup diskette BEFORE using!!!

PFS/PFS Report ---- 0-13..........ADDR=D5 AA 96
(REVISED)                     OVERIDE STANDARDIZER = Y
                              GAP BYTE 1=C0, GAP BYTE 2=D0
                              FILTER=C0-C8 (NO INVERSE)
    NOTE: Write protect before booting!!
PFS Graph --------- 0-22..........ADDR=D5 AA 96
                              OVERIDE STANDARDIZER = Y
                              GAP BYTE-1 = C0, GAP BYTE-2 = D0
                              FILTER = C0-C8 (NO INVERSE)

S O F T A P E:
Photar ------------ 0-22..........ADDR=D5 AA 96

S P E C I A L    D E L I V E R Y   S O F T W A R E:
Utopia Graphics --- 0-22..........ADDR=D5 AA 96
System                                   TURN ON 3.3 FILTER
                                   SECTMOD [F=16, C=ON, T=0, S=0]
```

```
                                     CHANGE ADDRESS 42 FROM 38 TO 18
Galactic Wars ----- 0-22..........ADDR=D5 AA 96
Bridge Tutor
Personal ---------- 0-22..........ADDR=D5 AA 96
  Finance Manager

S T O N E W A R E:
DB Master -------- 0-5...........ADDR=D5 AA 96, SYNC
Utility Pac #1        6.5-22.5.......SYNC

DB Master (old) --- 0-5...........ADDR=D5 AA 96
               6.5-22.5
DB Master (new) --- 0-5...........ADDR=D5 AA 96, SYNC
                6.5-22.5 SYNC

S T R A T E G I C   S I M U L A T I O N S:
Battle of Shiloh -- 0-22..........ADDR=D4 AA B7
Warp Factor

Cartels & --------- 0-0...........ADDR=D5 AA B5
  Cuthroats     2-22..........ADDR=DB D5 DE
Operation       1-1...........ADDR=D5 AA DA FF
  Apocalypse

Torpedo Fire ------ 0-22..........ADDR=D4 AA B7
Southern Command

S U B L O G I C:
FS-1 ------------- 0-0
                1.5-21 by 1.5..ADDR=DB AB BF
                           REDUCED ERROR CHECK
                7-8...........REDUCED ERROR CHECK
                9.5-9.5.......REDUCED ERROR CHECK

Saturn Navigator -- B-22..........ADDR=D5 AA FD, FIND MAX=08
                                (Errors on $11 and $17 OK)
                6.5-6.5.......FF FF D5 AA, FIND MAX=0C
                0-4...........ADDR=D5 AA B5
                11-11

Escape ----------- 0-22..........ADDR=D5 AA 96

A2-PB1 Pinball ---- 0-0...........ADDR=D5 AA 96, DATA MAX=25
                1-15..........ADDR=DB AB BF

S Y N E R G I S T I C   S O F T W A R E:
Escape from ------- 0-22..........ADDR=D5 AA 96, 'OVERIDE STANDARDIZER'
 Arcturus                            'OVERIDE NIBBLE FILTER'

S Y S T E M S    D E S I G N   L A B:
Gold Edition ------ 0-22..........ADDR=D5 AA 96
 Point Spread
 Prediction System

Win At The Races -- 0-22..........ADDR=D5 AA 96

S Y T O N I C    S O F T W A R E:
Interlude ----------0-22..........ADDR=D5 AA B5
```

```
T U R N K E Y    S O F T W A R E:
Ceiling Zero ------ 0-2............ADDR=D5 AA B5
                3-11...........ADDR=D6 AA B5
                               INS=DE AA EB F9, SYNC SIZ=0A

U N K N O W N    C O M P A N Y:
Magic Spells--------0-2  SYNC.......ADDR=D5 AA DB
                3-22 SYNC.......ADDR=D5 AA 96

U S A S O F T W A R E:
Apple World ------- 0-23
Star Dance -------- 0-22...........ADDR=D5 AA B5

V I D E X   C O R P
Pre-Boot System --- 0-22...........ADDR=D5 AA 96

V I S I C O R P:
Visicalc 3.3 ------ 0-0............ADDR=D5 AA 96
                2-22...........ADDR=D5 AA B5
                         (Errors toward end OK)

Visidex ----------- 0-22...........ADDR=D5 AA 96, Ins=DE AA EB FD
                                      SYNC SIZ=0A, FIX AMNT=04

Visiterm ---------- 0-22...........ADDR=D5 AA 96, Ins=DE AA EB FC
                                      SYNC SIZ=0A, FIX AMNT=04

Visitrend --------- 0-22...........ADDR=D5 AA 96, Ins=DE AA EB
  /Visiplot                           SYNC SIZ=0A, FIX AMNT=04

Desktop Plan II --- 0-22...........ADDR=D5 AA 96, Ins=AA EB FD
                                      SYNC SIZ=0A, FIX AMNT=04

Visifile ---------- 0-22...........ADDR=D5 AA 96, Ins=DE AA EB
                                      SYNC SIZ=0A, FIX AMNT=04

Visischedule------- 0-22...........ADDR=D5 AA 96, Ins=DE AA EB EC
                                      SYNC SIZ=0A, FIX AMNT=04

X P S:
Apple Cillin ------ 0-0............ADDR=D5 AA 96
                1-22...........ADDR=D5 AA B5

  11-11..........ADDR=D5 AA 96

==============================================================================
==============================================================================
DOCUMENT peekpoke.app
==============================================================================

+============================================================================+
!VER:2.1              (^)+=- PEEKS, POKES & CALLS -=+(^)        (c) May. 1984!
+============================================================================+
!Writen by:                \          for the APPLE ][+ & ][e W/DOS 3.3 & 48k!
!          -===THE=WIZARD==]>>>)}                                           !
!                          /            The World of Cryton: [414] 246-3965  !
+----------------------------------------------------------------------------+
                               SCROLLING WINDOW

POKE 32,L......Sets LEFT SIDE of the Scrolling Window {L=0 to 39}
POKE 33,W......Sets wI#
2$]he Scrolling Window {W=0 to 40-L}
POKE 34,T......Sets TOP of the Scrolling Window {T=0 to 23}
POKE 35,B......Sets BOTTOM of the Scrolling Window {B=0 to 23;B>T}

                            TEXT & CURSOR POSITION

POKE 36,CH.....Sets HORIZONTAL cursor position +1 {CH=0 to 39}
POKE 37,CV.....Sets VERTICAL cursor position +1 {CV=0 to 23}
CALL -1036.....MONITOR S/R to MOVE CURSOR RIGHT
CALL -1008.....MONITOR S/R to MOVE CURSOR LEFT
CALL -998......MONITOR S/R to MOVE CURSOR UP
CALL -990......MONITOR S/R PERFORM a VERTICAL TAB to ROW in ACCUMULATOR
CALL -980......MONITOR S/R PREFORM ESCAPE FUNCTION
CALL -958......CLEAR from CURSOR to END of PAGE {ESC-F}
CALL -936......MONITOR S/R HOME & CLEAR SCREEN {Destroys ACCUMULATOR & Y-REG}
CALL -926......MONITOR S/R PERFORM a CARRIAGE RETURN
CALL -922......MONITOR S/R PERFORM a LINE FEED
CALL -912......MONITOR S/R SCOLL UP 1 LINE {Destroys ACCUMULATOR & Y-REG}
CALL -868......MONITOR S/R CLEAR to END of LINE
CALL -868......CLEAR from CURSOR to END of LINE {ESC-E}
CALL -384......set INVERSE mode
CALL -380......set NORMAL mode

                               CHARACTER DISPLAY

POKE 50,255....White on Black {Normal}
POKE 50,63.....Black on White {Inverse}
POKE 50,127....Blinking {Flash}

                                 SCREEN FORMAT
                                   GRAPHICS

POKE -16304,0..Set Graphics display mode
POKE -16303,0..Set TEXT display mode
PEEK(-16358)...READ TEXT switch {If > 127 then it is "ON"}
POKE -16302,0..Set FULL-SCREEN Graphics display mode
POKE -16301,0..Set MIXED-SCREEN Graphics display mode
PEEK(-16357)...READ MIXED switch {If > 127 then it is "ON"}
POKE -16300,0..Turn page 2 HI-RES off {set page 1}
POKE -16299,0..Set display to HI-RES Graphics page 2
PEEK(-16356)...READ PAGE2 switch {If > 127 then it is "ON"}
POKE -16298,0..Turn HI-RES display mode off
POKE -16297,0..Set HI-RES Graphics display mode
PEEK(-16355)...READ HI-RES switch {If > 127 then it is "ON"}
CALL 62450.....CLEAR current HI-RES screen to BLACK
CALL 62454.....CLEAR current HI-RES screen to HCOLOR of last dot ploted
```

KEYBOARD

```
PEEK (-16384)..READ keyboard. If > 127 then a key was pressed. Always clear
               keyboard strobe before reading it.
POKE -16368,0..CLEARS the keyboard STROBE.
CALL -657......GET a LINE of input with NO PROMPT or LINE FEED, and wait.
CALL -662......GET a LINE of input with PROMPT, NO LINE FEED, and wait.
CALL -665......GET a LINE of input with PROMPT, LINE FEED, and wait.
  *NOTE: INPUT CHARACTERS are found in the INPUT BUFFER {Loc 512-767 $200-$2FF}
CALL -756......WAIT for KEY PRESS.
```

SOUND

```
X=PEEK(-16336).TOGGLES the SPEAKER {1 click}
POKE -16336,0..TOGGLES the SPEAKER {1 click (longer then PEEK)}
```

CASSETE

```
X=PEEK(-16352).TOGGLES CASSETTE OUTPUT once {1 click on cassette recording).
CALL -310......WRITE to TAPE
CALL -259......READ from TAPE
```

GAME PADDLES

```
PEEK(-16287)...READ PDL(0) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16286)...READ PDL(1) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16285)...READ PDL(2) BUTTON (SHIFT KEY) {If > 127 then switch is "ON"}
POKE -16296,1..CLEAR GAME I/O AN-0 OUTPUT {OFF-3.5V HIGH}
POKE -16295,0..SET GAME I/O AN-0 OUTPUT {ON-.3V LOW}
POKE -16294,1..CLEAR GAME I/O AN-1 OUTPUT {OFF-3.5V HIGH}
POKE -16293,0..SET GAME I/O AN-1 OUTPUT {ON-.3V LOW}
POKE -16292,1..CLEAR GAME I/O AN-2 OUTPUT {OFF-3.5V HIGH}
POKE -16291,0..SET GAME I/O AN-2 OUTPUT {ON-.3V LOW}
POKE -16290,1..CLEAR GAME I/O AN-3 OUTPUT {OFF-3.5V HIGH}
POKE -16289,0..SET GAME I/O AN-3 OUTPUT {ON-.3V LOW}
CALL -1250.....MONITOR S/R to READ PADDLE - X-Reg contains PDL # (0-3).
```

LO-RES GRAPHICS

```
CALL -2048.....PLOT a POINT {AC:Y-COORD  Y:X-COORD}
CALL -2023.....DRAW a HORIZONTAL LINE.
CALL -2008.....DRAW a VERTICAL LINE.
CALL -1998.....CLEAR LO-RES SCREEN 1 and set GRAPHICS mode.
CALL -1994.....CLEAR top 20 lines of LOW-RES Graphics
CALL -1977.....CALCULATE Graphics base ADDRESS.
CALL -1953.....INCREMENT COLOR by 2
CALL -1948.....ADJUST COLOR BYTE for both havles EQUAL.
CALL -1935.....MONITOR S/R to get SCREEN COLOR {AC:Y-COORD Y:X-COORD}
```

COLORS

0= Black	4= Dark Green	8= Brown	12= Green
1= Magenta	5= Grey	9= Orange	13= Yellow

2= Dark Blue	6= Medium Blue	10= Grey	14= Aqua
3= Light Purple	7= Light Blue	11= Pink	15= White

HI-RES GRAPHICS

```
POKE 800,H.....Set HORIZONTAL COORDINATE. H=MODULUS 256
POKE 801,H/256.H= 0 (left) to 279 (right)
               * Note: Both POKE 800 & 801 are required.
POKE 802,V.....Sets VERTICAL COORDINATE. {V= 0 (top) to 159 (bottom)}
POKE 804,S.....STARTING ADDRESS of SHAPE TABLE. S=MODULUS 256
POKE 805,S/256.Both 804 & 805 are required.
POKE 28,C......COLOR of SHAPE
POKE 812,x.....Sets COLOR for HI-RES
CALL -3805 PG..DRAWS predefind SHAPE.
CALL -3761.....PLOTS a POINT on the screen
CALL -3086.....Clear HI-RES screen to Black
CALL -3082.....Clear HI-RES screen to recent HCOLOR
CALL -2613.....HI-RES coordinates to ZERO page.
CALL -1438.....Pseudo-Reset
CALL -11780 M.."FIND" or POSITION
CALL -11272 S.."FIND" or BACKGROUND (HCOLOR 1 set for black background)
CALL -11471....HI-RES Graphics BACKGROUND (PAMAM=COLOR)
CALL -11462....HI-RES DRAW1(X0;Y0;COLOR)
CALL -11335....HI-RES SHLOAD
POKE 249,R.....Sets ROTATION of SHAPE {R=1 to 64; 0=Normal; 16=90' Clockwize}
PEEK (243).....FLASH MASK
PEEK (241).....SPEED (256 - current speed)
PEEK (234).....COLLISION COUNTER for shapes
PEEK (232-233).SHAPE TABLE starting address
POKE 231,S.....Sets SCALE of SHAPE
PEEK (230).....HI-RES PLOTING page. (32=Page 1, 64=Page 2, 96=Page 3)
PEEK (224-226).HI-RES GR X&Y Cordinates
POKE 228,x.....HI-RES GR COLOR BYTE (x can be 0-255)
```

HI-RES COLORS

0= Black1 {Gr/Vl}	1= Green	2= Violet	3= White1 {Gr/Vl}
4= Black2 {Or/Bl}	5= Orange	6= Blue	7= White2 {Or/Bl}

OTHER USEFULL CALLS
{Add +65536 to get pos. POKE's}

```
CALL 54915.....CLEARS STACK. Dose away with the false "OUT OF MEMORY" error.
CALL 1002......Reconnect DOS
CALL -8192.....RESET INTERGER BASIC. KILLS VARIABLES and CLEARS
CALL -8117.....LIST INTERGER PROGRAM
CALL -6739.....NEW
CALL -6729.....PLOTS a POINT on the screen
CALL -6090.....RUN INTERGER PROGRAM {SAVES VARIABLES}
CALL -4116.....RUN INTERGER PROGRAM {KILLS VARIABLES}
CALL -3973.....LOAD INTERGER PROGRAM from TAPE
CALL -3776.....SAVE INTERGER PROGRAM to TAPE
CALL -3774.....SAVE
CALL -3318.....CONTINUE
CALL -2458.....TURN ON MINI-ASSEMBLER
CALL -2423.....SWEET-16 INTERPRETER entry
CALL -q96.....MONITOR S/R DISASSEMBLER entry
CALL -1728.....MONITOR S/R-PRINT contents of X & Y {REG 9 as 4 HEX digits}
CALL -1716.....MONITOR S/R PRINT X BLANKS {X REG contains # to PRINT}
CALL -1402.....MONITOR S/R-IRQ HANDLER
CALL -1390.....MONITOR S/R-BREAK HNADLER
CALL -1370.....RE-BOOTS DISK SYSTEM
CALL -1321.....MONITOR S/R to display USER REGISTERS
CALL -1233.....MONITOR S/R SREEN INIT
CALL -1223.....MONITOR S/R set SCREEN to TEXT mode {Destroys ACCUMULATER}
CALL -1216.....MONITOR S/R set GRAPHICS mode {GR} {Destroys ACCUMULATER} CALL
CALL -1205.....MONITOR S/R set NORMAL WINDOW
CALL -1184.....Prints the 'Apple ][' at the top of your screen.
CALL -1181.....MONITOR S/R MULTIPLY ROUTINE
CALL -1148.....MONITOR S/R DIVIDE ROUTINE
CALL -1087.....MONITOR S/R CALCULATE TEXT BASE ADDRESS
CALL -1052.....MONITOR S/R SOUND BELL
CALL -1027.....MONITOR S/R OUTPUT A-REG as ASCII on TEXT SCREEN 1
CALL -856......MONITOR S/R WAIT LOOP
CALL -756......GET KEY from KEYBOARD {Destroys ACC & Y-REG} WAIT for KEY PRESS.
CALL -741......MONITOR S/R KEYIN ROUTINE
CALL -715......READ KEY & PERFORM ESCAPE FUNCTION if necessary.
CALL -678......Wait for RETURN
CALL -676......Bell; Wait or RETURN
CALL -670......PREFORM LINE CANCEL
CALL -665......PREFORM CARRIAGE RETURN & GET LINE of TEXT.
CALL -662......GET LINE of TEXT from KEYBOARD {X RETND with # of CHARACTERS}
CALL -657......INPUT; Accepts commas & collons.
               EX:PRINT "NAME (LAST, FIRST):";:CALL-657:A$="":FOR X= 512 TO 767
                  IF PEEK (X) < > 141 THEN A$= A$ + CHR$ (PEEK (X) -128) : NEXT
CALL -626......PRINT CARRIAGE RETURN {Destroys ACCUMULATOR & Y-REG}
CALL -622......PRINT A1H,A1L. Example: 10 POKE 60,A1H    20 POKE 61,A1L   30END
                ...Then RUN, CALL -622
CALL -550......PRINT CONTENTS of ACCUMULATOR. As 2 HEX DIGETS.
CALL -541......PRINT a HEX digit
CALL -531......OUTPUT CHARACTER IN ACCUMULATOR. {Destroys ACCUM. & Y-REG COUNT}
CALL -528......GET MONITOR CHARACTER OUTPUT
CALL -468......PERFORM MEMORY MOVE A1-A2 TO A4.
               Example:    10 POKE 60,LOB
                           20 POKE 61,HOB
                           30 POKE 62,LOE
                           40 POKE 63,HIE
                           50 POKE 66,LOD
                           60 POKE 67,HID
                ...Then RUN, CALL -468
                * Note: LOB is lo-byte of begining of memory to move, HIB is
                        high, LOE is low end, HIE is high, LOD is low destina-
                        tion, HID is high.
CALL -458......Perform MEMORY VERIFY (compare and list differences)
CALL -418......DISASSEMBLE 20 INSTRUCTIONS
CALL -415......DISASSEMBLER  Note: POKE start add. at 58-59 before calling.
CALL -378......set I FLAG
CALL -375......set KEYBOARD
CALL -336......JUMP to BASIC
CALL -333......CONTINUE BASIC
CALL -330......MEMORY LOCATION "GO"
CALL -321......DISPLAY A,S,Y,P,S REG. {CURRENT VALUES}
CALL -318......PERFORM MONITOR TRACE
CALL -307......WRITE OUT cassette tape
CALL -259......READ FROM cassette tape {LIMITS A1 to A2}
CALL -211......PRINT "ERR" & SOUNDS BELL {Destroys ACCUMULATOR & Y-REG}
CALL -198......PRINT BELL {Destroys ACCUMULATOR & Y-REG}
CALL -193......MONITOR & SWEET-16 "RESTORE"
CALL -188......MONITOR "RESTR1"
CALL -182......MONITOR & SWEET-16 "SAVE"
CALL -180......MONITOR "SAV1"
CALL -167......ENTER MONITOR RESET, TEXT mode, "COLD START"
CALL -155......ENTER MONITOR, ring BELL, "WARM START"
CALL -151......Go to MONITOR
CALL -144......SCAN INPUT BUFFER {ADDRESS $200...}
               EX: A$ = "300:A9 C1 20 ED FD 18 69 01 C9 DB D0 F6 60 300G D823G"
                   FOR X=1 TO LEN(A$): POKE 511+X,ASC (MID$ (A$,X,1))+128: NEXT
                   POKE 72,0: CALL -144
```

ERRORS

```
POKE 216,0.....RESETS ERROR FLAG
PEEK (216).....If = 127 then an ERROR was detected.
PEEK (212).....Returns ERROR CODE FLAG in decimal.
```

MEMORY ALLOCATION

RANGE	USE DESCRIPTION
$0-$1FF	Program wook space {not for USER}
$200-$2FF	Keyboard Character buffer
$300-$3FF	Available for short Machine langauge rutine
$400-$7FF	Screen display page 1 TEXT or GR
$800-$1FFF	Available RAM for BASIC programs
$2000-$3FFF	HGR page 1
$4000-$5FFF	HGR page 2
$6000-$95FF	Available RAM for BASIC programs
$9600-$9CFF	DOS files buffers {Maxfiles 3}
$9D00-$AAFC	Main DOS routines
$AAFD-$B7B4	File Manager
$B7B5-$BFFF	RWTS
$C000-$CFFF	I/O Hardware {end of RAM}
$D000-$FFFF	ROM {I/O Addresses}

SPECIAL MEMORY LOCATIONS

LOCATION	USE DESCRIPTION
$18	First track of data {for DOS}
$19	First sector of data {for DOS}
$1A	Number of sectors to load {for DOS}
$1B	The HIGH BYTE of the buffer {LO is always 00} {DOS Command}
$1A - $1B	Shape pointer used by DRAW and XDRAW
$1C	Last color used {HCOLOR converted to its color byte}
$26 - $27	Address of byte contained X,Y point
$2B	Boot SLOT * 16
$2C	Lo-res line END-point
$30	COLOR * 17
$33	Prompt-Char, {POKE 51,0:GOTO line #; Defeats NOT DIRECT COMMAND}
$68	LOMEM: {LOW BYTE is always 00}
$4E - $4F	Random - Number feild
$69 - $6A	Simple Variables
$6B - $6C	Start of ARRAY - Space
$6D - $6E	END of ARRAY - Space

```
$6F - $70  ! Start of STRING storage
$73 - $74  ! HIMEM: $73=LO BYTE
$75 - $76  ! Line # being executed
$77 - $78  ! Line # where program stopped
$79 - $7A  ! Address of executing line #
$7B - $7C  ! Current DATA line #
$7D - $7E  ! Next DATA address
$7F - $80  ! Input or Data address
$81 - $82  ! Last used Variable NAME: VAR$ = CHR$(PEEK(129)) + CHR$(PEEK(130))
$83 - $84  ! Last used variable address
$AF - $B0  ! End of Applesoft program
$D8        ! ONERR flag   NOTE: POKE 216,0 cancels ONERR function
$DA - $DB  ! Line # of ONERR error
$DE        ! ONERR error code {Dec. PEEK (222)}
$E0 - $E1  ! X-coordinate (0-279) in HEX {Low,High}
$E2        ! Y-coordinate (0-191) in HEX
$E4        ! Color being used {0=0:42=1:85=2:127=3:128=4:170=5:213=6:255=7}
$E6        ! Current HI-RES page being used {$20: Page one, $40: Page two}
$E7        ! Current SCALE (0-256)
$E8 - $U9
jk6Xtion of shape table {Low,High}
$EA        ! Collision counter {used by XDRAW and DRAW}
$3D0 - $3D2! JUMP vector to DOS Warmstart {JMP $9DBF}
$3D3 - $3D5! JUMP vector to DOS Coldstart {JMP $9D84}
$3D6 - $3D8! JUMP vector to DOS File Manager {JMP $AAFD}
$3D9 - $3DB! JUMP vector to RWTS {JMP $B7B5}
$3DC - $3E2! Subroutine to locate File Manager PARM list {LDA $9D0F;LDY $9D0E}
$3E3 - $3E9! Subroutine to locate RWTS PARM list {LDA $AAC2; LDY $AAC1; RTS}
$3EA - $3EE! JUMP to replace DOS intercepts subroutine {JMP $A851; NOP; NOP}
$3EF - $3F1! JUMP vector to Autostart BRK Handler {JMP $FA59}
$3F2 - $3F3! Autostart Reset handler {$9DBF}
$3F4       ! POWER-UP byte ($3F3 EOR $A5) {$38}
$3F5 - $3F7! JUMP vector to Applesoft & Handler {JMP $FF58}
$3F8 - $3FA! JUMP vector to CTR-Y handler {JMP $FF65}
$3FB - $3FD! JUMP vector to NMI handler {JMP $FF65}
$3FE - $3FF! Vector for IRQ handler {$FF65}
$AA61.$AA60! LENGTH of file just loaded {$AA61 is the HIGH BYTE}
$AA73.$AA72! STARTING ADDRESS of file just loaded {$AA73 is the HIGH BYTE}
$FBB3      ! SIGNATURE byte {$06 = //e  :  $EA = ][+}
```

MISCELLANEOUS INFORMATION
CONTROL RESET

To make it run your program type this:

```
      10 POKE 1010,102
      20 POKE 1011,213
      30 POKE 1012,112
```

To make it send you to MONITOR type this:

```
      POKE 1010,105
      POKE 1011,255
      CALL -1169
```

To make it BOOT DOS type this:

```
      POKE 592,0
      POKE 1012,0
```

* Note: The origanal values are:

```
      PEEK(592) = 255  DivDos64k  Norml
      PEEK(1010)= 3       60       191
      PEEK(1011)= 224     191      157
      PEEK(1012)= 69      26       56
```

VERY QUICK SORTING ROUTINE

```
1000 FOR I = 1 TO N - 1 : REM N = # OF ITEMS
1010 P = I
1020 FOR J = I + 1 TO N
1030 IF A(J) < A(P) THEN P = J
1040 NEXT J
1050 T = A(I) : A(I) = A(P) : A(P) = T
1060 NEXT I
```

DOS MEMORY LOCATIONS

```
LOCATION    ! USE DESCRIPTION
------------+-----------------------------------------------------------------+
$3D0 - $3D2 ! Re-enter DOS Vector
$3F2 - $3F4 ! Reset Vector   EX: POKE 1012,0 Reboots  {Norm: 56}
$3F5 - $3F7 ! Ampersand Vector. EX: POKE 1014,165:POKE 1015,214 -=> LIST
            !                   EX: POKE 1014,110:POKE 1015,165 -=> CATALOG
            !                   EX: POKE 1014,18 :POKE 1015,217 -=> RUN
$3F8 - $3FA ! Crtl - Y Vector
$A56E       ! Catalog Routine.  Also  CALL 42350
$9E42       ! Greeting program RUN-FLAG {POKE 40514,X: 52=BRUN, 20=EXEC}
$A884-$A907 ! DOS Commands
$A972-$AA3E ! ERROR messages
$A960-$AA61 ! Last BLOAD Lenght {LEN = PEEK (43616) + PEEK (43617) * 256}
$AA72-$AA73 ! Last BLOAD START {STR = PEEK (43634) + PEEK (43635) * 256}
$AA57       ! MAX Files Values
$AAB1       ! Max files Default Value
$AA68       ! Drive - Number  EX: POKE 43624,DR   DR= Drive for I/O
$AA6A       ! Slot - Number
$AC01       ! Catalog Track number.
$AE17       ! # Characters -1 in catalog file name.
$B3A7-$B3AE ! File type codes
$B3AF-$B3BA ! Disk Vol. Heading
$B3C1       ! Disk Vol. Number
$B3F0       ! Number of Sectors per Track
```

DOS MISCELLANEOUS

```
To defeat the "NOT DIRECT COMMAND" error type: POKE 51,0 : GOTO line #
To kill the INIT command do: Poke 42309,96  or  $A545:60
To kill the INIT command in normal DOS type: POKE 42309,96  or $A545:60
If you want a basic program to load in after HGR {more memory than LOMEM:16384}
   use this loader program:
         10 POKE 16384,0 : POKE 104,64 : REM STARTING LOCATION OF PROGRAM
         20 PRINT CHR$(4) "RUN PROGRAM"

  *Note:To put things back to normal use this program:
             10 POKE 2048,0 : POKE 104,8
             20 PRINT CHR$(4) "RUN OLD PROGRAM"

If PEEK(-18070) = 150 then your using DOS 3.3 .
```

```
POKE 40193,PEEK(40193)-N:CALL 42964...Move DOS buffers down N*256 bytes.
POKE 44452,N+1:POKE 44605,N...........Allows N file names before Catalog pause.
POKE 44505,234:POKE 44579,234:POKE 44580,234...Cancels return after file names.
POKE 44578,234:POKE 44579,234:POKE 44580,234...Cancels catalog pause.
POKE 44599,234:POKE 44600,234.........Wait for key input after every file name.

Here are some POKEs that turn on the DRIVES but do not READ or WRITE.  These
   can be used as scare tactics. EX. PRINT "INITIALIZING DISK" : POKE -16151,0
      POKE -16151,0........TURNS ON DRIVE 1
      POKE -16135,0........TURNS ON DRIVE 2
      POKE -16152,0........TURNS OFF DRIVE 1
      POKE -16136,0........TURNS OFF DRIVE 2

To stop CATALOG for a key input after every file name type:
      POKE 44599,234 {NORM 208}
      POKE 44600,234 {NORM 8}

To omit the pause after a full screen of CATALOG then type:
      $AE34:60  or  POKE 44569,96

For WILDCARD DOS files useing "=" type: (from monitor)
      B201:4C 71 BA
      BA69:E8 B1 42 DD C6 B4 D0 0A C8 C0 1E D0 F3
           AE 9C B3 18 60 C9 AD F0 F7 4C 0B B2
```

MISCELLANEOUS

```
To make the program in memory run when any Syntax (but DOS commands) is typed
   then put this line in:  10 POKE 214,128 {Norm 0}

PEEK (104).....If 8 is returned then APPLESOFT is in ROM. Any other value means
               APPLESOFT is in RAM or not available.
POKE 2049,1....Repeatedly LISTs first line of program.
CALL -856......TIME DELAY. POKE 69,XX to set amount of delay.
CALL -1182.....Prints the Apple ][ across the top of your screen.

POKE 49107,234:POKE 49108,234:POKE 49109,234...Prevents language card re-load.

For "true" random number generation use RND(PEEK(78)+PEEK(79)*256).

POKE 1912+SLOT,1 on APPLE PARALLEL CARD (with P1-02 PROM) will enable LINEFEED.
POKE 1912+SLOT,0 on APPLE PARALLEL CARD (with P1-02 PROM) => disable LINEFEEDS.

REMAINDER {Mod} type: R = X - (INT (X / Y) * Y)
To ROUND to N digets past the decimal type: X = INT (X * (10^N) +.5) / (10^N)
QUADRATIC formula : R1 = (-B + SQR (B^2 - 4 * A * C)) / (2 * A)
                    R2 = (-B - SQR (B^2 - 4 * A * C)) / (2 * A)
```

CONVERSIONS

```
To change VOLUME # xxx to SECTORS FREE = xxx then type the following:
      ADC0:20 69 BA
      BB69:A9 00 85 40 85 41 A0 C8 18 B9 F2 B3 F0 0E 0A 90 FB 48 E6 40 D0 02 E6
           41 68 18 90 F0 88 D0 E9 A6 40 A5 41 20 24 ED 60
      B3AF:A0 BD A0 D4 C3 C5 D3 A0 C5 C5 D2 C6

If you own a //e then you can get the functions of an 80 col card (save 80 col)
   with out the card. Just type: POKE 49162,0  then type: PR#3
```

```
+==============================================================================+
!  If you find an error or want to add something, please leave me a message!  !
+==============================================================================+
add something, please leave me a message!  !
+================================================================
```

==
DOCUMENT peeks.pokes
==

Applesoft: PEEKs, POKEs, and CALLs To make Applesoft programs read data from memory, write data to memory, or pass control to machine language programs, programmers use Applesoft's PEEK, POKE, and CALL statements. Here is an explanation of each statement's function.

PEEK makes a program read a memory location. The format of the statement is PEEK (<memory location>) where <memory location> is a positive integer from 0 to 65535. Programmers use PEEK most commonly with a variable: X% = PEEK (2048) assigns the value located at 2048 to the integer variable X%.

POKE makes a program write a value to a memory location. The format of the statement is POKE <memory location>,<value> where <memory location> is a positive integer from 0 to 65535 and <value> is a positive integer from 0 to 255. Programmers use POKE most commonly to write data directly to memory: POKE 2048,128 assigns the value 128 to the memory location 2048.

CALL makes a program pass control to a machine language routine at some memory location. The format of the statement is CALL <memory location> where <memory location> is a positive or negative integer from -32768 to 32767 or a positive integer from 0 to 65535 (note that the signed integers from -32768 to 32767 represent exactly the same memory locations as the positive integers from 0 to 65535). Programmers use CALL most commonly used to invoke routines built into the Apple II's ROM. For example, the statement CALL -936 invokes the routine which clears the screen and homes the cursor (just like using Applesoft's HOME statement).

To change the screen display or make sounds and other special effects on the Apple II, Apple II Plus, Apple IIe, Apple IIc and Apple IIGS, Applesoft accesses various memory locations. Each particular CPU's reference manual includes a memory map where you can find the segments of memory used by text, graphics, Applesoft, the monitor and peripheral cards.

Apple-published memory locations remain the same for most members of the Apple II family; other internal locations may change. Therefore, to assure that your programs will work properly on all Apple II family computers, do not use entry points other than those printed in the Apple manuals.

Locations used to communicate with interface cards may be found in the manuals for those devices. For example, memory locations used by the Apple 80-column card are found in the 80-Column Text Card Manual and the Extended 80-Column Text Card Supplement.

Many computer and book stores sell books with listings of Applesoft, monitor ROM, DOS 3.3 and ProDOS memory locations. You may find the following publications useful:

--What's Where in the Apple by William Luebbert; Micro Ink.
--Beneath Apple DOS by Don Wirth and Pieter Lechner; Quality Software.
--Beneath Apple ProDOS by Don Wirth and Pieter Lechner; Quality Software.
--The Apple Almanac by Eric Goez and Williams Sanders; Datamost, Inc.

Apple Technical Communications

```
=============================================================================
DOCUMENT peeks.pokes.1
=============================================================================

1)  CAN  POKE AT 1024-2039 THE ASCII VALUE OF A DIGIT AND PUT THAT DIGIT
ON THE CRT. (HEX AT $400).  1024 IS UPPER LEFT  CORNER,  2039  IS  LOWER
RIGHT.

2) & JUMPS TO MEMORY ADDRESS $3F4, WORKS LIKE A CALL TO THAT ADDRESS.

3)
POKE 32,33,34,35 FOR TXT WINDOW.
POKE 36,X TO TAB PRINTER (X= ONE LESS THEN SPACES TO TAB.
POKE 37,X SET CURSON VERT POSITION.
POKE 50,(255=NORMAL 63=INVERSE 127=FLASH).
POKE 54,X (CSWL) USER CHAR OUT.
POKE 55,X (CSWH) USER CHAR OUT.
POKE 56,X (KSWL) USER CHAR IN VECTOR.
POKE 57.X (KSWL) USER CHAR IN VECTOR.
POKE 212,128 TO TURN APPLESOFT INTO RUN ONLY MODE.
POKE 216,0 CANCEL ONERR.
POKE 243,X WHERE X=1-255 WILL USUALLY MAKE LISTINGS UN-READABLE.
POKE 1014,10:POKE1015,165 WILL CATLOG YOUR DISK WHEN THE '&' KEY IS PRESSED.
POKE 1014,165:POKE 1015,214 WILL MAKE THE '&' KEY LIST PROGRAM.
POKE 2049,1 TO MAKE FIRST LINE OF PROGRAM LIST REPEATEDLY.
POKE -16151 TURNS ON DRIVE 1, POKE -16152 TURNS OFF DRIVE 1.
POKE -16135 TURNS ON DRIVE 2, POKE -16316 TURNS OFF DRIVE 2.
POKE -16289,0 SETS GAME AN #3
POKE -16290,0 CLEARS GAME AN #3.
POKE -16291,0 SETS GAME AN #2.
POKE -16292,0 CLEARS GAME AN #2.
POKE -16293,0 SETS GAME AN #1.
POKE -16294,0 CLEARS GAME AN #1.
POKE -16295,0 SETS GAME AN #0.
POKE :16296,0 CLEARS GAME AN #0.
POKE -16297,0 FOR HGR
POKE -16298,0 FOR LOW GR
POKE -16299 PAGE 2, POKE -16300 PAGE 1
POKE -16300,0 CLEAR PAGE 2.
POKE -16301 MIXED TEXT AND GRAPHICS.
POKE -16302 ALL GRAPHICS
POKE -16303 TEXT
POKE -16304 GRAPHICS
POKE -16368 CLEARS KEYBOARD STROBE.
POKE -21912,X TO SELECT DISKDRIVE WITHOUT EXECUTING A COMMAND.
POKE 43697,X TO SET THE MAXFILES DEFAULT. NOTE-0 MAY BOMB DOS.
POKE 43698,X WHERE X=ASCII OF DOS  CMND CHARACTER.(NORMALLY A CNTRL D.
POKE 44505,234 THEN POKE 44506,234 THIS SHOWS DELETED FILES IN CATALOG.
     NORMALIZE BY POKEING IN 48,74 AT SAME LOCATIONS.
POKE -49167 TURN ON ALTERNATE CHR. SET APPLE IIE. POKE 49166 TO TURN OFF.

4)
PEEK(36) READ CURSOR HORZ POSITION (0-39)
PEEK (37) READS CURSOR VERT POSITION (0-23).
PEEK(74)+PEEK(75)*256 CURRENT LOMEM
PEEK(76)+PEEK(77)*256 CURRENT INTEGER HIMEM.
PEEK (103) + PEEK(104) * 256 IS THE BEGINNIG ADDRESS OF FP PROGRAMS.
PEEK (104) IF VALUE = 8 THEN APPLESOFT IN ROM, IF NOT 8, THEN
     APPLESOFT IN RAM OR MEMORY.
PEEK(115)+PEEK(116)*256 IS CURRENT APPLESOFT HIMEM.
PEEK(175)+PEEK(176)*256 POINTS TO APPLESOFT PROGRAM END.
PEEK (202) + PEEK (203) * 256 IS BEGINNING ADDRESS OF INT PROGRAMS.
PEEK (218)+PEEK(219)*256 WILL PRINT APPLESOFT LINE WHERE LAST ERROR
     OCCURED IF 'ONERR' SET.
PEEK (222) GIVES ERROR CODE ON ONERR MESSAGE.
PEEK (225) + PEEK (225) * 256 IS HORIZONTAL POSITION OF LAST HPLOT.
PEEK (226) IS VERTICAL POSITION OF LAST HPLOT.
PEEK (232) + PEEK (233) * 256 IS BEGINNING ADDRESS OF SHAPE TABLE.
PEEK (-16284) PADDLE (3) BUTTON
PEEK (-16285) PADDLE (2) BUTTON
PEEK (-16286) PADDLE (1) BUTTON
PEEK (-16287) PADDLE (0) BUTTON, ALL BUTTONS ARE > 127 IF DEPRESSED.
PEEK (-16336) WHEN DEFINED AS VARIABLE, CLICKS SPEAKER.
PEEK (-16352) TOGGLES THE CASSETTE OUTPUT.
PEEK (-16368) READS KEYBOARD.
PEEK (-16384) READS KEYBOARD. IF >127 THEN KEY WAS HIT.
PEEK (-18070) IF VALUE IS 150 THEN IN DOS 3.3
PEEK (43616) + (43617) : 256 IS LENGTH OF BIN PROGRAM.
PEEK (43634) + (43635) * 256 IS BEGINNING ADDRESS OF BIN PROGRAM.
PEEK (46064) RETURN NUMBER OF SECTORS ON DISK. 16= DOS3.3.

5)  CALLS

CALL -144 SCANS INPUT BUFFER
CALL -151 ENTER MONITOR
CALL -155 ENTER MONITOR WITH BELL.
CALL -167 ENTER MONITOR AND RESET
CALL -198 RING BELL
CALL -259 READ FROM TAPE
CALL -310 WRITE TO TAPE
CALL -321 DISPLAY THE A, X, Y, P, AND S REGISTERS.
CALL -380 SET NORMAL DISPLAY MODE.
CALL -384 SET INVERSE DISPLAY MODE
CALL -458 VERIFY (COMPARE AND LIST DIFFERENCES)
CALL -550 PRINT HEX VALUE OF ACCUMULATOR
CALL -657 ALLOWS A LINE OF INPUT (WITH COMMAS AND COLONS) AND NO LINE FEED
          OR PROMPT.
CALL -662 GET LINE OF INPUT WITH PROMPT, NO LINEFEED
CALL -665 GET LINE OF INPUT WITH PROMPT, LINEFEED
CALL -678 WAIT TILL RETURN IS PRESSED
CALL -670 PERFORM A LINE CANCEL.
CALL -756 WAIT TILL ANY KEY IS PRESSED
CALL -856 TIME DELAY. POKE 69,XX TO SET AMOUNT OF DELAY.
CALL -868 CLEAR FROM CURSOR TO END OF LINE.
CALL -875 CLEAR WHOLE LINE OF TEXT
CALL -912 SCROLL ONE LINE.
CALL -922 CARRAIGE RETURN
CALL -936 CLEAR SCREEN AND HOME CURSOR.
CALL -958 CLEAR TEXT FROM CURSOR TO BOTTOM OF SCREEN
CALL -998 MOVE CURSOR UP ONE LINE
CALL -1002 RECONNECT DOS HOOKS.
CALL -1008 MOVE CURSOR LEFT ONE SPACE
CALL -1036 MOVE CURSOR RIGHT ONE SPACE
CALL -1184 CLEAR SCREEN AND PRINT APPLE LOGO.
CALL -1216 SET GR TT CALL -1370 BOOT DISK
CALL -1401 BOOT DISK (UNCOMMON)
CALL -1728 DISPLAY HEX VALUES OF X AND Y REGISTOR
CALL -1953 CHANGE COLOR BY 3.
CALL -1994 CLEAR VIDIO SCREEN.
CALL -1998 CLEAR GRAPHIC SCREEN.
CALL -2458 ENTER MINI-ASSEMBLER
CALL -3305 RESUME FROM APPLESOFT ERROR.
CALL -3106 HGR2
CALL -3116 HGR1
CALL -3318 EXEC INT BASIC 'CON' CMD.
CALL -3722 TURN OFF INT BASIC TRACE.
CALL -3727 TURN ON INT BASIC TRACE.
CALL -3776 SAVE INTEGER TO TAPE.
CALL -3973 LOAD INTEGER PROGRAM FROM TAPE.
CALL -6090 RUN INTEGER PROGRAM.
CALL -8117 LIST INTEGER PROGRAM.
CALL -8192 END INTERER AND KILL PROGRAM.
CALL -9382 OUTPUT A '?'.
CALL -9385 OUTPUT A SPACE.
CALL -9477 OUTPUT A CARRAGE RETURN.
CALL -9582 PRINTS CATALOG.
CALL -42350 CATALOG
CALL -54915 DOES AWAY WITH 'OUT OF MEM ERROR' WHEN MEMORY STILL LEFT.
CALL 62450 CLEAR HIRES SCREEN TO BLACK.
CALL 62454 CLEAR HIRES SCREEN TO HCOLOR LAST HPLOTTED.
```

6) ELIMINATE PAUSE IN CATALOG: GO TO MONITOR AND TYPE AE34:60

7) IF YOU HAVE LANGUAGE/MEMORY CARD: READ TRACK 0, SECTOR $09. BYTE $CC IS $81, CHANGE TO $10. NOW WHEN YOUR DO PR#6, WHATEVER WAS IN MEMORY CARD, STAYS THERE.

8) FOR TRUE RANDOM NUMBER: USE RND(PEEK(78)+PEEK(79)*256) IN APPLESOFT PROGRAM.

9) MAKE PROGRAM LISTINGS INTO GARBAGE BY DOING A POKE 33,90.

10) POKE 50,250 OR 50,127 AND WATCH WHAT HAPPENS.

11) APPLE PARRALLEL CARD WITH P1-02 PROMS. POKE 1912+SLOT,1 TO ENABLE LINE FEED. POKE 1912+SLOT,0 TO DISABLE LINE FEED.

12) DEFEAT 'NOT DIRECT COMMAND' ERROR WHEN TRYING TO RESUME A PROGRAM FROM COMMAND MODE BY: POKE 51,128:GOTOX (X = LINE # TO GO TO).

```
=================================================================================
DOCUMENT peeks.pokes.2
=================================================================================

The following is a list of peeks & pokes in the zero page area this
list was obtained from a beagle bros chart...
Decimal | Hexadecimal |
---------------------------------------------------------------------------------
32      | $20         | Text window left-edge (0-39)
33      | $21         | Text window width (1-40)
34      | $22         | Text window top-edge (0-23)
35      | $23         | Text window bottom (1-24)
36      | $24         | Horizontal cursor-position (0-39)
37      | $25         | Vertical cursor-position (0-23)
43      | $2B         | Boot slot * 16 (after boot only)
44      | $2C         | Lo-res line end-point
48      | $30         | Lo-res COLOR * 17
50      | $32         | Text output format [63=INVERSE 255=NORMAL 127=FLASH]
51      | $33         | Prompt-character (NOTE: POKE 51,0:GOTO LINE # will
        |             | sometimes prevent a false NOT DIRECT COMMAND obtained
        |             | with GOTO # alone.)
74-75   | $4A-$4B     | LOMEM address (INT)
76-77   | $4C-$4D     | HIMEM address (INT)
78-79   | $4E-$4F     | Random-Number Field
103-104 | $67-$68     | Start of Applesoft program (NOTE: FP sets start of  a
        | program to normal 2049.  NOTE: To load a program another location=LOC
        | POKE 103,LOC-INT(LOC/256)*256:POKE 104,INT(LOC/256):POKE LOC,0
105-106 | $69-$6A     | LOMEM (Start of varible space & end of Applesoft program
107-108 | $6B-$6C     | Start of array space  (FP)
109-110 | $6D-$6E     | End of array space  (FP)
111-112 | $6F-$70     | Start of string-storage  (FP)
115-116 | $73-$74     | HIMEM (NOTE: HIMEM-1 is the highest Applesoft address.)
117-118 | $75-$76     | Line# being executed.  (FP)
119-120 | $77-$78     | Line# where program stopped.  (FP)
121-122 | $79-$7A     | Address of line executing.  (FP)
123-124 | $7B-$7C     | Current DATA line#
125-126 | $7D-$7E     | Next DATA address
127-128 | $7F-$80     | INPUT or DATA address
129-130 | $81-$82     | Var.last used. VAR$=CHR$(PEEK(129))+CHR$(PEEK(130)) (FP)
131-132 | $83-$84     | Last-Used-Varible Address  (FP)
175-176 | $AF-$B0     | End of Applesoft Program (Normally=LOMEM)
202-203 | $CA-$CB     | Start of Program Address (INT)
204-205 | $CC-CD      | End of Varible Storage (INT)
214     | $D6         | RUN Flag (POKE 214,255 turns Applesoft into run only.)
216     | $D8         | ONERR Flag (POKE 216,0 cancels ONERR)
218-219 | $DA-$DB     | Line# of ONERR Error
222 _ _ | $DE _ _ _ _ | ONERR Error Codes (DOS Errors have no ?)
_-_-_-_-_-_-_-_-_-_-_-| 0=?NEXT WITHOUT FOR          16=?SYNTAX ERROR
_-_-_-_-_-_-_-_-_-_-_-| 1=LANGUAGE NOT AVAIBLE       22=?RETURN WITHOUT GOSUB
_-_-_-_-_-_-_-_-_-_-_-| 2/3=RANGE ERROR              42=?OUT OF DATA
_-_-_-_-_-_-_-_-_-_-_-| 4=WRITE-PROTECTED            53=?ILLEGAL QUANTITY
_-_-_-_-_-_-_-_-_-_-_-| 5=END OF DATA                69=?OVERFLOW
_-_-_-_-_-_-_-_-_-_-_-| 6=FILE NOT FOUND             77=?OUT OF MEMORY
_-_-_-_-_-_-_-_-_-_-_-| 7=VOLUME MISMATCH            90=?UNDER'D STATEMENT
_-_-_-_-_-_-_-_-_-_-_-| 8=I/O ERROR                 107=?BAD SUBSCRIPT
_-_-_-_-_-_-_-_-_-_-_-| 9=DISK FULL                 120=?REDIM'D ARRAY
_-_-_-_-_-_-_-_-_-_-_-| 10=FILE LOCKED              133=?DIVISION BY ZERO
```

```
_-_-_-_-_-_-_-_-_-_-_-| 11=SYNTAX ERROR            163=?TYPE MISMATCH
_-_-_-_-_-_-_-_-_-_-_-| 12=NO BUFFERS AVAILABLE    176=?STRING TOO LONG
_-_-_-_-_-_-_-_-_-_-_-| 13=FILE TYPE MISMATCH      191=?FORMULA TOO COMPLEX
_-_-_-_-_-_-_-_-_-_-_-| 14=PROGRAM TO LARGE        224=?UDEF'D FUNCTION
_-_-_-_-_-_-_-_-_-_-_-| 15=NOT DIRECT COMMAND      254=?REENTER
_-_-_-_-_-_-_-_-_-_-_-|                            255=CTRL-C INTERRUPT
224-225 | $E0.E1      | X Coordinate of last HPLOT (0-279)
226     | $E2         | Y Coordinate of last HPLOT (0-191)
228     | $E4         | HCOLOR codes:0=0 42=1 85=2 127=3 128=4 170=5 213=6 255=6
230     | $E6         | Hi res plotting page (32=page 1, 64=page 2, 96=page 3)
231     | $E7         | SCALE (NOTE:SCALE=0 is equivalent to a SCALE of 256.)
232-233 | $E8.E9      | Shape table start address.
234     | $EA         | Hi-Res Collision-Check (IF PEEK(234)=0 then the shape
        |             | started at a non-black hi-res point.)
241     | $F1         | SPEED (NOTE:PEEK(241) is 256 minus the current SPEED.)
243     | $F3         | FLASH Mask
249     | $F9         | ROT
```

```
=============================================================================
DOCUMENT peeks.pokes.3.1
=============================================================================

+==========================================================================+
!VER:2.1            (^)+=- PEEKS, POKES & CALLS -=+(^)      (c) May. 1984!
+==========================================================================+
!Writen by:              \        for the APPLE ][+ & ][e W/DOS 3.3 & 48k!
!          -===THE=ENFORCER]>>>)}                                         !
!                        /          The World of Cryton: [414] 246-3965  !
+--------------------------------------------------------------------------+

                             SCROLLING WINDOW

POKE 32,L......Sets LEFT SIDE of the Scrolling Window {L=0 to 39}
POKE 33,W......Sets WIDTH of the Scrolling Window {W=0 to 40-L}
POKE 34,T......Sets TOP of the Scrolling Window {T=0 to 23}
POKE 35,B......Sets BOTTOM of the Scrolling Window {B=0 to 23;B>T}

                          TEXT & CURSOR POSITION

POKE 36,CH.....Sets HORIZONTAL cursor position +1 {CH=0 to 39}
POKE 37,CV.....Sets VERTICAL cursor position +1 {CV=0 to 23}
CALL -1036.....MONITOR S/R to MOVE CURSOR RIGHT
CALL -1008.....MONITOR S/R to MOVE CURSOR LEFT
CALL -998......MONITOR S/R to MOVE CURSOR UP
CALL -990......MONITOR S/R PERFORM a VERTICAL TAB to ROW in ACCUMULATOR
CALL -980......MONITOR S/R PREFORM ESCAPE FUNCTION
CALL -958......CLEAR from CURSOR to END of PAGE {ESC-F}
CALL -936......MONITOR S/R HOME & CLEAR SCREEN {Destroys ACCUMULATOR & Y-REG}
CALL -926......MONITOR S/R PERFORM a CARRIAGE RETURN
CALL -922......MONITOR S/R PERFORM a LINE FEED
CALL -912......MONITOR S/R SCOLL UP 1 LINE {Destroys ACCUMULATOR & Y-REG}
CALL -868......MONITOR S/R CLEAR to END of LINE
CALL -868......CLEAR from CURSOR to END of LINE {ESC-E}
CALL -384......set INVERSE mode
CALL -380......set NORMAL mode

                             CHARACTER DISPLAY

POKE 50,255....White on Black {Normal}
POKE 50,63.....Black on White {Inverse}
POKE 50,127....Blinking {Flash}

                               SCREEN FORMAT
                                 GRAPHICS

POKE -16304,0..Set Graphics display mode
POKE -16303,0..Set TEXT display mode
PEEK(-16358)...READ TEXT switch {If > 127 then it is "ON"}
POKE -16302,0..Set FULL-SCREEN Graphics display mode
POKE -16301,0..Set MIXED-SCREEN Graphics display mode
PEEK(-16357)...READ MIXED switch {If > 127 then it is "ON"}
POKE -16300,0..Turn page 2 HI-RES off {set page 1}
POKE -16299,0..Set display to HI-RES Graphics page 2
PEEK(-16356)...READ PAGE2 switch {If > 127 then it is "ON"}
POKE -16298,0..Turn HI-RES display mode off
POKE -16297,0..Set HI-RES Graphics display mode
PEEK(-16355)...READ HI-RES switch {If > 127 then it is "ON"}
CALL 62450.....CLEAR current HI-RES screen to BLACK
CALL 62454.....CLEAR current HI-RES screen to HCOLOR of last dot ploted
```

KEYBOARD

```
PEEK (-16384)..READ keyboard. If > 127 then a key was pressed. Always clear
               keyboard strobe before reading it.
POKE -16368,0..CLEARS the keyboard STROBE.
CALL -657......GET a LINE of input with NO PROMPT or LINE FEED, and wait.
CALL -662......GET a LINE of input with PROMPT, NO LINE FEED, and wait.
CALL -665......GET a LINE of input with PROMPT, LINE FEED, and wait.
 *NOTE: INPUT CHARACTERS are found in the INPUT BUFFER {Loc 512-767 $200-$2FF}
CALL -756......WAIT for KEY PRESS.
```

SOUND

```
X=PEEK(-16336).TOGGLES the SPEAKER {1 click}
POKE -16336,0..TOGGLES the SPEAKER {1 click (longer then PEEK)}
```

CASSETE

```
X=PEEK(-16352).TOGGLES CASSETTE OUTPUT once {1 click on cassette recording).
CALL -310......WRITE to TAPE
CALL -259......READ from TAPE
```

GAME PADDLES

```
PEEK(-16287)...READ PDL(0) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16286)...READ PDL(1) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16285)...READ PDL(2) BUTTON (SHIFT KEY) {If > 127 then switch is "ON"}
POKE -16296,1..CLEAR GAME I/O AN-0 OUTPUT {OFF-3.5V HIGH}
POKE -16295,0..SET GAME I/O AN-0 OUTPUT {ON-.3V LOW}
POKE -16294,1..CLEAR GAME I/O AN-1 OUTPUT {OFF-3.5V HIGH}
POKE -16293,0..SET GAME I/O AN-1 OUTPUT {ON-.3V LOW}
POKE -16292,1..CLEAR GAME I/O AN-2 OUTPUT {OFF-3.5V HIGH}
POKE -16291,0..SET GAME I/O AN-2 OUTPUT {ON-.3V LOW}
POKE -16290,1..CLEAR GAME I/O AN-3 OUTPUT {OFF-3.5V HIGH}
POKE -16289,0..SET GAME I/O AN-3 OUTPUT {ON-.3V LOW}
CALL -1250.....MONITOR S/R to READ PADDLE - X-Reg contains PDL # (0-3).
```

LO-RES GRAPHICS

```
CALL -2048.....PLOT a POINT {AC:Y-COORD  Y:X-COORD}
CALL -2023.....DRAW a HORIZONTAL LINE.
CALL -2008.....DRAW a VERTICAL LINE.
CALL -1998.....CLEAR LO-RES SCREEN 1 and set GRAPHICS mode.
CALL -1994.....CLEAR top 20 lines of LOW-RES Graphics
CALL -1977.....CALCULATE Graphics base ADDRESS.
CALL -1953.....INCREMENT COLOR by 2
CALL -1948.....ADJUST COLOR BYTE for both havles EQUAL.
CALL -1935.....MONITOR S/R to get SCREEN COLOR {AC:Y-COORD Y:X-COORD}
```

COLORS

0= Black	4= Dark Green	8= Brown	12= Green
1= Magenta	5= Grey	9= Orange	13= Yellow

```
2= Dark Blue       6= Medium Blue      10= Grey            14= Aqua
3= Light Purple    7= Light Blue       11= Pink            15= White
```

HI-RES GRAPHICS

```
POKE 800,H.....Set HORIZONTAL COORDINATE. H=MODULUS 256
POKE 801,H/256.H= 0 (left) to 279 (right)
               * Note: Both POKE 800 & 801 are required.
POKE 802,V.....Sets VERTICAL COORDINATE. {V= 0 (top) to 159 (bottom)}
POKE 804,S.....STARTING ADDRESS of SHAPE TABLE. S=MODULUS 256
POKE 805,S/256.Both 804 & 805 are required.
POKE 28,C......COLOR of SHAPE
POKE 812,x.....Sets COLOR for HI-RES
CALL -3805 PG..DRAWS predefind SHAPE.
CALL -3761.....PLOTS a POINT on the screen
CALL -3086.....Clear HI-RES screen to Black
CALL -3082.....Clear HI-RES screen to recent HCOLOR
CALL -2613.....HI-RES coordinates to ZERO page.
CALL -1438.....Pseudo-Reset
CALL -11780 M.."FIND" or POSITION
CALL -11272 S.."FIND" or BACKGROUND (HCOLOR 1 set for black background)
CALL -11471....HI-RES Graphics BACKGROUND (PAMAM=COLOR)
CALL -11462....HI-RES DRAW1(X0;Y0;COLOR)
CALL -11335....HI-RES SHLOAD
POKE 249,R.....Sets ROTATION of SHAPE {R=1 to 64; 0=Normal; 16=90' Clockwize}
PEEK (243).....FLASH MASK
PEEK (241).....SPEED (256 - current speed)
PEEK (234).....COLLISION COUNTER for shapes
PEEK (232-233).SHAPE TABLE starting address
POKE 231,S.....Sets SCALE of SHAPE
PEEK (230).....HI-RES PLOTING page. (32=Page 1, 64=Page 2, 96=Page 3)
PEEK (224-226).HI-RES GR X&Y Cordinates
POKE 228,x.....HI-RES GR COLOR BYTE (x can be 0-255)
```

HI-RES COLORS

```
0= Black1 {Gr/Vl}  1= Green            2= Violet           3= White1 {Gr/Vl}
4= Black2 {Or/Bl}  5= Orange           6= Blue             7= White2 {Or/Bl}
```

OTHER USEFULL CALLS
{Add +65536 to get pos. POKE's}

```
CALL 54915.....CLEARS STACK. Does away with the false "OUT OF MEMORY" error.
CALL 1002......Reconnect DOS
CALL -8192.....RESET INTERGER BASIC. KILLS VARIABLES and CLEARS
CALL -8117.....LIST INTERGER PROGRAM
CALL -6739.....NEW
CALL -6729.....PLOTS a POINT on the screen
CALL -6090.....RUN INTERGER PROGRAM {SAVES VARIABLES}
CALL -4116.....RUN INTERGER PROGRAM {KILLS VARIABLES}
CALL -3973.....LOAD INTERGER PROGRAM from TAPE
CALL -3776.....SAVE INTERGER PROGRAM to TAPE
CALL -3774.....SAVE
CALL -3318.....CONTINUE
CALL -2458.....TURN ON MINI-ASSEMBLER
CALL -2423.....SWEET-16 INTERPRETER entry
CALL -1906.....MONITOR S/R DISASSEMBLER entry
CALL -1728.....MONITOR S/R-PRINT contents of X & Y {REG 9 as 4 HEX digits}
CALL -1716.....MONITOR S/R PRINT X BLANKS {X REG contains # to PRINT}
CALL -1402.....MONITOR S/R-IRQ HANDLER
CALL -1390.....MONITOR S/R-BREAK HANDLER
CALL -1370.....RE-BOOTS DISK SYSTEM
CALL -1321.....MONITOR S/R to display USER REGISTERS
CALL -1233.....MONITOR S/R SREEN INIT
CALL -1223.....MONITOR S/R set SCREEN to TEXT mode {Destroys ACCUMULATER}
CALL -1216.....MONITOR S/R set GRAPHICS mode {GR} {Destroys ACCUMULATER} CALL
CALL -1205.....MONITOR S/R set NORMAL WINDOW
CALL -1184.....Prints the 'Apple ][' at the top of your screen.
CALL -1181.....MONITOR S/R MULTIPLY ROUTINE
CALL -1148.....MONITOR S/R DIVIDE ROUTINE
CALL -1087.....MONITOR S/R CALCULATE TEXT BASE ADDRESS
CALL -1052.....MONITOR S/R SOUND BELL
CALL -1027.....MONITOR S/R OUTPUT A-REG as ASCII on TEXT SCREEN 1
CALL -856......MONITOR S/R WAIT LOOP
CALL -756......GET KEY from KEYBOARD {Destroys A & Y-REG} WAIT for KEY PRESS.
CALL -741......MONITOR S/R KEYIN ROUTINE
CALL -715......READ KEY & PERFORM ESCAPE FUNCTION if necessary.
CALL -678......Wait for RETURN
CALL -676......Bell; Wait or RETURN
CALL -670......PREFORM LINE CANCEL
CALL -665......PREFORM CARRIAGE RETURN & GET LINE of TEXT.
CALL -662......GET LINE of TEXT from KEYBOARD {X RETND with # of CHARACTERS}
CALL -657......INPUT; Accepts commas & collons.
              EX:PRINT "NAME (LAST, FIRST):";:CALL-657:A$="":FOR X= 512 TO 767
                 IF PEEK (X) < > 141 THEN A$= A$ + CHR$ (PEEK (X) -128) : NEXT
CALL -626......PRINT CARRIAGE RETURN {Destroys ACCUMULATOR & Y-REG}
CALL -622......PRINT A1H,A1L. Example: 10 POKE 60,A1H  20 POKE 61,A1L   30END
                ...Then RUN, CALL -622
CALL -550......PRINT CONTENTS of ACCUMULATOR. As 2 HEX DIGETS.
CALL -541......PRINT a HEX digit
CALL -531......OUTPUT CHARACTER IN ACCUMULATOR. {Destroys A & Y-REG COUNT}
CALL -528......GET MONITOR CHARACTER OUTPUT
CALL -468......PERFORM MEMORY MOVE A1-A2 TO A4.
               Example:    10 POKE 60,LOB
                           20 POKE 61,HOB
                           30 POKE 62,LOE
                           40 POKE 63,HIE
                           50 POKE 66,LOD
                           60 POKE 67,HID
                ...Then RUN, CALL -468

* Note: LOB is lo-byte of begining of memory to move, HIB is
high, LOE is low end, HIE is high, LOD is low destination, HID is high.

CALL -458......Perform MEMORY VERIFY (compare and list differences)
CALL -418......DISASSEMBLE 20 INSTRUCTIONS
CALL -415......DISASSEMBLER  Note: POKE start add. at 58-59 before calling.
CALL -378......set I FLAG
CALL -375......set KEYBOARD
CALL -336......JUMP to BASIC
CALL -333......CONTINUE BASIC
CALL -330......MEMORY LOCATION "GO"
CALL -321......DISPLAY A,S,Y,P,S REG. {CURRENT VALUES}
CALL -318......PERFORM MONITOR TRACE
CALL -307......WRITE OUT cassette tape
CALL -259......READ FROM cassette tape {LIMITS A1 to A2}
CALL -211......PRINT "ERR" & SOUNDS BELL {Destroys ACCUMULATOR & Y-REG}
CALL -198......PRINT BELL {Destroys ACCUMULATOR & Y-REG}
CALL -193......MONITOR & SWEET-16 "RESTORE"
CALL -188......MONITOR "RESTR1"
CALL -182......MONITOR & SWEET-16 "SAVE"
CALL -180......MONITOR "SAV1"
CALL -167......ENTER MONITOR RESET, TEXT mode, "COLD START"
CALL -155......ENTER MONITOR, ring BELL, "WARM START"
CALL -151......Go to MONITOR
CALL -144......SCAN INPUT BUFFER {ADDRESS $200...}
             EX: A$ = "300:A9 C1 20 ED FD 18 69 01 C9 DB D0 F6 60 300G D823G"
                 FOR X=1 TO LEN(A$): POKE 511+X,ASC (MID$ (A$,X,1))+128: NEXT
                 POKE 72,0: CALL -144
```

ERRORS

```
POKE 216,0.....RESETS ERROR FLAG
PEEK (216).....If = 127 then an ERROR was detected.
PEEK (212).....Returns ERROR CODE FLAG in decimal.
```

MEMORY ALLOCATION

```
RANGE        ! USE DESCRIPTION
------------+---------------------------------------------+
$0-$1FF      ! Program wook space {not for USER}
$200-$2FF    ! Keyboard Character buffer
$300-$3FF    ! Available for short Machine langauge rutine
$400-$7FF    ! Screen display page 1 TEXT or GR
$800-$1FFF   ! Available RAM for BASIC programs
$2000-$3FFF  ! HGR page 1
$4000-$5FFF  ! HGR page 2
$6000-$95FF  ! Available RAM for BASIC programs
$9600-$9CFF  ! DOS files buffers {Maxfiles 3}
$9D00-$AAFC  ! Main DOS routines
$AAFD-$B7B4  ! File Manager
$B7B5-$BFFF  ! RWTS
$C000-$CFFF  ! I/O Hardware {end of RAM}
$D000-$FFFF  ! ROM {I/O Addresses}
```

SPECIAL MEMORY LOCATIONS

```
LOCATION   ! USE DESCRIPTION
----------+-----------------------------------------------------------------+
$18        ! First track of data {for DOS}
$19        ! First sector of data {for DOS}
$1A        ! Number of sectors to load {for DOS}
$1B        ! The HIGH BYTE of the buffer {LO is always 00} {DOS Command}
$1A - $1B  ! Shape pointer used by DRAW and XDRAW
$1C        ! Last color used {HCOLOR converted to its color byte}
$26 - $27  ! Address of byte contained X,Y point
$2B        ! Boot SLOT * 16
$2C        ! Lo-res line END-point
$30        ! COLOR * 17
$33        ! Prompt-Char, {POKE 51,0:GOTO line #; Defeats NOT DIRECT COMMAND}
$68        ! LOMEM: {LOW BYTE is always 00}
$4E - $4F  ! Random - Number feild
$69 - $6A  ! Simple Variables
$6B - $6C  ! Start of ARRAY - Space
```

```
$6D - $6E  ! END of ARRAY - Space
$6F - $70  ! Start of STRING storage
$73 - $74  ! HIMEM: $73=LO BYTE
$75 - $76  ! Line # being executed
$77 - $78  ! Line # where program stopped
$79 - $7A  ! Address of executing line #
$7B - $7C  ! Current DATA line #
$7D - $7E  ! Next DATA address
$7F - $80  ! Input or Data address
$81 - $82  ! Last used Variable NAME: VAR$ = CHR$(PEEK(129)) + CHR$(PEEK(130))
$83 - $84  ! Last used variable address
$AF - $B0  ! End of Applesoft program
$D8        ! ONERR flag   NOTE: POKE 216,0 cancels ONERR function
$DA - $DB  ! Line # of ONERR error
$DE        ! ONERR error code {Dec. PEEK (222)}
$E0 - $E1  ! X-coordinate (0-279) in HEX {Low,High}
$E2        ! Y-coordinate (0-191) in HEX
$E4        ! Color being used {0=0:42=1:85=2:127=3:128=4:170=5:213=6:255=7}
$E6        ! Current HI-RES page being used {$20: Page one, $40: Page two}
$E7        ! Current SCALE (0-256)
$E8 - $E9  ! Location of shape table {Low,High}
$EA        ! Collision counter {used by XDRAW and DRAW}
$3D0 - $3D2! JUMP vector to DOS Warmstart {JMP $9DBF}
$3D3 - $3D5! JUMP vector to DOS Coldstart {JMP $9D84}
$3D6 - $3D8! JUMP vector to DOS File Manager {JMP $AAFD}
$3D9 - $3DB! JUMP vector to RWTS {JMP $B7B5}
$3DC - $3E2! Subroutine to locate File Manager PARM list {LDA $9D0F;LDY $9D0E}
$3E3 - $3E9! Subroutine to locate RWTS PARM list {LDA $AAC2; LDY $AAC1; RTS}
$3EA - $3EE! JUMP to replace DOS intercepts subroutine {JMP $A851; NOP; NOP}
$3EF - $3F1! JUMP vector to Autostart BRK Handler {JMP $FA59}
$3F2 - $3F3! Autostart Reset handler {$9DBF}
$3F4       ! POWER-UP byte ($3F3 EOR $A5) {$38}
$3F5 - $3F7! JUMP vector to Applesoft & Handler {JMP $FF58}
$3F8 - $3FA! JUMP vector to CTR-Y handler {JMP $FF65}
$3FB - $3FD! JUMP vector to NMI handler {JMP $FF65}
$3FE - $3FF! Vector for IRQ handler {$FF65}
$AA61.$AA60! LENGTH of file just loaded {$AA61 is the HIGH BYTE}
$AA73.$AA72! STARTING ADDRESS of file just loaded {$AA73 is the HIGH BYTE}
$FBB3      ! SIGNATURE byte {$06 = //e  :  $EA = ][+}
```

```
==============================================================================
DOCUMENT peeks.pokes.3.2
==============================================================================
```

MISCELLANEOUS INFORMATION
CONTROL RESET

To make it run your program type this:

```
      10 POKE 1010,102
      20 POKE 1011,213
      30 POKE 1012,112
```

To make it send you to MONITOR type this:

```
      POKE 1010,105
      POKE 1011,255
      CALL -1169
```

To make it BOOT DOS type this:

```
      POKE 592,0
      POKE 1012,0
```

* Note: The origanal values are:

```
      PEEK(592) = 255  DivDos64k  Norml
      PEEK(1010)= 3       60       191
      PEEK(1011)= 224     191      157
      PEEK(1012)= 69      26       56
```

VERY QUICK SORTING ROUTINE

```
1000 FOR I = 1 TO N - 1 : REM N = # OF ITEMS
1010 P = I
1020 FOR J = I + 1 TO N
1030 IF A(J) < A(P) THEN P = J
1040 NEXT J
1050 T = A(I) : A(I) = A(P) : A(P) = T
1060 NEXT I
```

DOS MEMORY LOCATIONS

```
LOCATION    ! USE DESCRIPTION
------------+---------------------------------------------------------------+
$3D0 - $3D2 ! Re-enter DOS Vector
$3F2 - $3F4 ! Reset Vector   EX: POKE 1012,0 Reboots  {Norm: 56}
$3F5 - $3F7 ! Ampersand Vector. EX: POKE 1014,165:POKE 1015,214 -=> LIST
            !                   EX: POKE 1014,110:POKE 1015,165 -=> CATALOG
            !                   EX: POKE 1014,18 :POKE 1015,217 -=> RUN
$3F8 - $3FA ! Crtl - Y Vector
$A56E       ! Catalog Routine.  Also  CALL 42350
$9E42       ! Greeting program RUN-FLAG {POKE 40514,X: 52=BRUN, 20=EXEC}
$A884-$A907 ! DOS Commands
$A972-$AA3E ! ERROR messages
$A960-$AA61 ! Last BLOAD Lenght {LEN = PEEK (43616) + PEEK (43617) * 256}
$AA72-$AA73 ! Last BLOAD START {STR = PEEK (43634) + PEEK (43635) * 256}
$AA57       ! MAX Files Values
$AAB1       ! Max files Default Value
$AA68       ! Drive - Number  EX: POKE 43624,DR   DR= Drive for I/O
$AA6A       ! Slot - Number
```

```
$AC01       ! Catalog Track number.
$AE17       ! # Characters -1 in catalog file name.
$B3A7-$B3AE ! File type codes
$B3AF-$B3BA ! Disk Vol. Heading
$B3C1       ! Disk Vol. Number
$B3F0       ! Number of Sectors per Track
```

DOS MISCELLANEOUS

```
To defeat the "NOT DIRECT COMMAND" error type: POKE 51,0 : GOTO line #
To kill the INIT command do: Poke 42309,96  or  $A545:60
To kill the INIT command in normal DOS type: POKE 42309,96  or $A545:60
If you want a basic program to load in after HGR
   {more memory than LOMEM:16384}  use this loader program:

         10 POKE 16384,0 : POKE 104,64 : REM STARTING LOCATION OF PROGRAM
         20 PRINT CHR$(4) "RUN PROGRAM"

  *Note:To put things back to normal use this program:
             10 POKE 2048,0 : POKE 104,8
             20 PRINT CHR$(4) "RUN OLD PROGRAM"

If PEEK(-18070) = 150 then your using DOS 3.3 .

POKE 40193,PEEK(40193)-N:CALL 42964...Move DOS buffers down N*256 bytes.
POKE 44452,N+1:POKE 44605,N...........Allows N file names before Catalog pause
POKE 44505,234:POKE 44579,234:POKE 44580,234...Cancels return after file names
POKE 44578,234:POKE 44579,234:POKE 44580,234...Cancels catalog pause.
POKE 44599,234:POKE 44600,234.........Wait for key input after every file name

Here are some POKEs that turn on the DRIVES but do not READ or WRITE.  These
can be used as scare tactics. EX. PRINT "INITIALIZING DISK" : POKE -16151,0

      POKE -16151,0........TURNS ON DRIVE 1
      POKE -16135,0........TURNS ON DRIVE 2
      POKE -16152,0........TURNS OFF DRIVE 1
      POKE -16136,0........TURNS OFF DRIVE 2

To stop CATALOG for a key input after every file name type:
      POKE 44599,234 {NORM 208}
      POKE 44600,234 {NORM 8}

To omit the pause after a full screen of CATALOG then type:
      $AE34:60  or  POKE 44569,96

For WILDCARD DOS files useing "=" type: (from monitor)
      B201:4C 71 BA
      BA69:E8 B1 42 DD C6 B4 D0 0A C8 C0 1E D0 F3
           AE 9C B3 18 60 C9 AD F0 F7 4C 0B B2
```

MISCELLANEOUS

```
To make the program in memory run when any Syntax (but DOS commands) is typed
   then put this line in:  10 POKE 214,128 {Norm 0}

PEEK (104).....If 8 is returned then APPLESOFT is in ROM. Any other value
                 means APPLESOFT is in RAM or not available.
POKE 2049,1....Repeatedly LISTs first line of program.
```

```
CALL -856......TIME DELAY. POKE 69,XX to set amount of delay.
CALL -1182.....Prints the Apple ][ across the top of your screen.

POKE 49107,234:POKE 49108,234:POKE 49109,234...Prevents language card re-load.

For "true" random number generation use RND(PEEK(78)+PEEK(79)*256).

POKE 1912+SLOT,1 on APPLE PARALLEL CARD (with P1-02 PROM) will enable LINEFEED
POKE 1912+SLOT,0 on APPLE PARALLEL CARD (with P1-02 PROM) => disable LINEFEEDS

REMAINDER {Mod} type: R = X - (INT (X / Y) * Y)
To ROUND to N digets past the decimal type: X = INT (X * (10^N) +.5) / (10^N)
QUADRATIC formula : R1 = (-B + SQR (B^2 - 4 * A * C)) / (2 * A)
                    R2 = (-B - SQR (B^2 - 4 * A * C)) / (2 * A)

                                    CONVERSIONS

To change VOLUME # xxx to SECTORS FREE = xxx then type the following:
   ADC0:20 69 BA
   BB69:A9 00 85 40 85 41 A0 C8 18 B9 F2 B3 F0 0E 0A 90 FB 48 E6 40 D0 02 E6
        41 68 18 90 F0 88 D0 E9 A6 40 A5 41 20 24 ED 60
   B3AF:A0 BD A0 D4 C3 C5 D3 A0 C5 C5 D2 C6

If you own a //e then you can get the functions of an 80 col card
(save 80 col) with out the card. Just type: POKE 49162,0  then type: PR#3

 -END-
```

```
==============================================================================
DOCUMENT pitfall2.txt
==============================================================================

3

-Press (SPACE BAR) to quit-

SOFT-DOCS FOR PITFALL II:  LOST CAVERNS
OBJECT:
     Help Pitfall Harry find his niece Rhonda, the cowardly cat
Quickclaw, and the great Raj diamond.  On the way, grab all the
white gold bars you can (and watch out for athe pesky stone-aged
rat).  There is no time limit in the caverns!

STARTING OUT:
     -Hit 'ESC' to start the game from the title page.
     -Sound is toggled by hitting ctrl-S.

JOYSTICK CONTROLS:
     -To move Pitfall Harry left or right, move controller left
or right.
     -To jump, press button 0.
     -To decend a ladder, pull controller back just before Harry
reaches the hole.  To ascend a ladder, push controller forward.
     -To catch a balloon, push the left button to jump.  Move
controller left or right to float left or right.  To speed up,
push controller forward; to slow down, pull controller back.

KEYBOARD CONTROLS:
     -On the IIe, use the four arrow keys to move a direction.
     -On the II+, use the I,M,J,K keys to move in a direction.
     -Press the SPACE BAR to jump.
     -Press the shift key to stop Pitfall Harry.
     -To speed up a balloon, press the UP arrow.  To slow down,
press the DOWN arrow.  (If you don't have up and down arrows,
give it up: get a joystick.  Control-keys suck.)

DANGERS:
     Keep away from frogs, bats, condors, eels, and albino
scorpions.  Touching any of them will set you back!

WHITE CROSSES:
     Whenever Pitfall Harry succumbs to a danger, he is
magically transported back to the last white cross he touched.
So, be sure and touch each of these mystical Incan healing
centers as you encounter them.

REWARDS:
     You start out with 4000 points.  Thereafter, you receive:

     -5000 points for every white gold bar.
     -15000 points for the cave rat.
     -20000 points for the Raj diamond
     -10000 points for Rhonda
     -10000 points for Quickclaw
```

     Touching dangerous creatures or falling unintentionally
loses points.

<END OF FILE>

(?=Database Menu, Q=Quit) Read #}i

```
==============================================================================
DOCUMENT pm2600.app
==============================================================================
```

-Another article from Sir Briggs of SCDW-
Hey all you phreakers! Bet you didn't know about this! It's...

The Poor Man's 2600 Hertz!!

What the hell could I be talking about!?!? Well, let's say you're really hard up (not in your usual sense, this time). You really need to make 2600 Hertz so you can have lotsa phun on the trunk lines, right? But your mom and dad didn't give you a blue box for Christmas- just an Apple! And of course you don't have a nice precision music card (like mine) or an Apple Cat. So what the hell can you do? Well, you're not out of it yet. You, too, can make 2600 Hertz! Yes, that's right! With NO additional hardware! Try and beat that with a stick (or your fist even for that matter). And I bet you've even figured out that I'm about to tell you just how to do this. Well, you're right! EVERYBODY KNOWS... that at $FCA8, there's a little routine called "WAIT". We are going to use that to produce the needed delay in the production of our tone. Yes, you will have to use a little machine language. But I'm going to show you exactly what to type here. So even you, yes YOU Poindexter, can get this right! Here's all you do...

If you have an Apple //e with the enhancement installed, just type CALL-151 from BASIC and get into the monitor. From there, hit a "!" to use the mini-assembler. Enter this exactly as it appears...

```
!1000: LDX $C030
! LDA #$06
! JSR $FCA8
! JMP $1000
```

And there you have it! Hit <RET> to get back to the monitor. Then, type "1000G" and listen to that beautiful tone! Not EXACTLY 2600 Hz, but close enough to do the trick!

For you non-enhanced types, you can just load up INTEGER BASIC (Ha!) and type "F666G" from the monitor and use the mini-assembler there. After typing the above code in, type "$FF69G" to return to the monitor, and proceed as above. You would do that on a][+, too (people still use those!?).

In all cases, just hit RESET to shut the thing up! Use it as you will. In case you didn't know, you can use that tone to reset SPRINT, MCI, etc. nodes to there dial tone. That way, you don't have to keep punching in your local number first. Just type the code and go! Pretty nice. Well, you can learn what to do from all the philes around about blue boxing. 2600 Hz doesn't work on 800 numbers here anymore. SHIT! What's going on? ESS? Well, if you live in ESS, don't try this! They'll snag your little butt fer sher! Then it's off to reform school for you! Well, have phun! And remember...

I didn't tell you this!

Sir Briggs of the SouthCentral Discount Waremeisters of Texas A & M

We brought you:

```
     AE: TAC 1.1
     Scream--> The Ultimate Telephone Terrorizer
     Duo-Disk Modz

Be on the lookout for Scream 2.0, The ALF Box (for those with ALF Music
Synthesizer Cards), a one-pass copier for Apple Extended Memory Cards,

                       and MUCH, MUCH MORE!

  BYE!
```

```
==============================================================================
DOCUMENT pokelist.app
==============================================================================

+=============================================================================+
!VER:2.1           (^)+=- PEEKS, POKES & CALLS -=+(^)  (c) May. 1984!
+=============================================================================+
!Writen by:           \        for the APPLE ][+ & ][e W/DOS 3.3 & 48k!
!          -===THE=WIZARD==]>>>)}                             !
!                     /        The World of Cryton: [414] 246-3965  !
+-----------------------------------------------------------------------------+
                         SCROLLING WINDOW

POKE 32,L......Sets LEFT SIDE of the Scrolling Window {L=0 to 39}
POKE 33,W......Sets WIDTH of the Scrolling Window {W=0 to 40-L}
POKE 34,T......Sets TOP of the Scrolling Window {T=0 to 23}
POKE 35,B......Sets BOTTOM of the Scrolling Window {B=0 to 23;B>T}

                      TEXT & CURSOR POSITION

POKE 36,CH.....Sets HORIZONTAL cursor position +1 {CH=0 to 39}
POKE 37,CV.....Sets VERTICAL cursor position +1 {CV=0 to 23}
CALL -1036.....MONITOR S/R to MOVE CURSOR RIGHT
CALL -1008.....MONITOR S/R to MOVE CURSOR LEFT
CALL -998......MONITOR S/R to MOVE CURSOR UP
CALL -990......MONITOR S/R PERFORM a VERTICAL TAB to ROW in ACCUMULATOR
CALL -980......MONITOR S/R PREFORM ESCAPE FUNCTION
CALL -958......CLEAR from CURSOR to END of PAGE {ESC-F}
CALL -936......MONITOR S/R HOME & CLEAR SCREEN {Destroys ACCUMULATOR & Y-REG}
CALL -926......MONITOR S/R PERFORM a CARRIAGE RETURN
CALL -922......MONITOR S/R PERFORM a LINE FEED
CALL -912......MONITOR S/R SCOLL UP 1 LINE {Destroys ACCUMULATOR & Y-REG}
CALL -868......MONITOR S/R CLEAR to END of LINE
CALL -868......CLEAR from CURSOR to END of LINE {ESC-E}
CALL -384......set INVERSE mode
CALL -380......set NORMAL mode

                         CHARACTER DISPLAY

POKE 50,255....White on Black {Normal}
POKE 50,63.....Black on White {Inverse}
POKE 50,127....Blinking {Flash}

                         SCREEN FORMAT
                           GRAPHICS

POKE -16304,0..Set Graphics display mode
POKE -16303,0..Set TEXT display mode
PEEK(-16358)...READ TEXT switch {If > 127 then it is "ON"}
POKE -16302,0..Set FULL-SCREEN Graphics display mode
POKE -16301,0..Set MIXED-SCREEN Graphics display mode
PEEK(-16357)...READ MIXED switch {If > 127 then it is "ON"}
POKE -16300,0..Turn page 2 HI-RES off {set page 1}
POKE -16299,0..Set display to HI-RES Graphics page 2
PEEK(-16356)...READ PAGE2 switch {If > 127 then it is "ON"}
POKE -16298,0..Turn HI-RES display mode off
POKE -16297,0..Set HI-RES Graphics display mode
```

```
PEEK(-16355)...READ HI-RES switch {If > 127 then it is "ON"}
CALL 62450.....CLEAR current HI-RES screen to BLACK
CALL 62454.....CLEAR current HI-RES screen to HCOLOR of last dot ploted
```

KEYBOARD

```
PEEK (-16384)..READ keyboard. If > 127 then a key was pressed. Always clear
             keyboard strobe before reading it.
POKE -16368,0..CLEARS the keyboard STROBE.
CALL -657......GET a LINE of input with NO PROMPT or LINE FEED, and wait.
CALL -662......GET a LINE of input with PROMPT, NO LINE FEED, and wait.
CALL -665......GET a LINE of input with PROMPT, LINE FEED, and wait.
  *NOTE: INPUT CHARACTERS are found in the INPUT BUFFER {Loc 512-767 $200-$2FF}
CALL -756......WAIT for KEY PRESS.
```

SOUND

```
X=PEEK(-16336).TOGGLES the SPEAKER {1 click}
POKE -16336,0..TOGGLES the SPEAKER {1 click (longer then PEEK)}
```

CASSETE

```
X=PEEK(-16352).TOGGLES CASSETTE OUTPUT once {1 click on cassette recording).
CALL -310......WRITE to TAPE
CALL -259......READ from TAPE
```

GAME PADDLES

```
PEEK(-16287)...READ PDL(0) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16286)...READ PDL(1) push BUTTON switch {If > 127 then switch is "ON"}
PEEK(-16285)...READ PDL(2) BUTTON (SHIFT KEY) {If > 127 then switch is "ON"}
POKE -16296,1..CLEAR GAME I/O AN-0 OUTPUT {OFF-3.5V HIGH}
POKE -16295,0..SET GAME I/O AN-0 OUTPUT {ON-.3V LOW}
POKE -16294,1..CLEAR GAME I/O AN-1 OUTPUT {OFF-3.5V HIGH}
POKE -16293,0..SET GAME I/O AN-1 OUTPUT {ON-.3V LOW}
POKE -16292,1..CLEAR GAME I/O AN-2 OUTPUT {OFF-3.5V HIGH}
POKE -16291,0..SET GAME I/O AN-2 OUTPUT {ON-.3V LOW}
POKE -16290,1..CLEAR GAME I/O AN-3 OUTPUT {OFF-3.5V HIGH}
POKE -16289,0..SET GAME I/O AN-3 OUTPUT {ON-.3V LOW}
CALL -1250.....MONITOR S/R to READ PADDLE - X-Reg contains PDL # (0-3).
```

LO-RES GRAPHICS

```
CALL -2048.....PLOT a POINT {AC:Y-COORD  Y:X-COORD}
CALL -2023.....DRAW a HORIZONTAL LINE.
CALL -2008.....DRAW a VERTICAL LINE.
CALL -1998.....CLEAR LO-RES SCREEN 1 and set GRAPHICS mode.
CALL -1994.....CLEAR top 20 lines of LOW-RES Graphics
CALL -1977.....CALCULATE Graphics base ADDRESS.
CALL -1953.....INCREMENT COLOR by 2
CALL -1948.....ADJUST COLOR BYTE for both havles EQUAL.
CALL -1935.....MONITOR S/R to get SCREEN COLOR {AC:
Y-COORD Y:X-COORD}
```

COLORS

0= Black	4= Dark Green	8= Brown	12= Green
1= Magenta	5= Grey	9= Orange	13= Yellow

```
2= Dark Blue         6= Medium Blue      10= Grey        14= Aqua
3= Light Purple    7= Light Blue       11= Pink          15= White
```

HI-RES GRAPHICS

```
POKE 800,H.....Set HORIZONTAL COORDINATE. H=MODULUS 256
POKE 801,H/256.H= 0 (left) to 279 (right)
             * Note: Both POKE 800 & 801 are required.
POKE 802,V.....Sets VERTICAL COORDINATE. {V= 0 (top) to 159 (bottom)}
POKE 804,S.....STARTING ADDRESS of SHAPE TABLE. S=MODULUS 256
POKE 805,S/256.Both 804 & 805 are required.
POKE 28,C......COLOR of SHAPE
POKE 812,x.....Sets COLOR for HI-RES
CALL -3805 PG..DRAWS predefind SHAPE.
CALL -3761.....PLOTS a POINT on the screen
CALL -3086.....Clear HI-RES screen to Black
CALL -3082.....Clear HI-RES screen to recent HCOLOR
CALL -2613.....HI-RES coordinates to ZERO page.
CALL -1438.....Pseudo-Reset
CALL -11780 M.."FIND" or POSITION
CALL -11272 S.."FIND" or BACKGROUND (HCOLOR 1 set for black background)
CALL -11471....HI-RES Graphics BACKGROUND (PAMAM=COLOR)
CALL -11462....HI-RES DRAW1(X0;Y0;COLOR)
CALL -11335....HI-RES SHLOAD
POKE 249,R.....Sets ROTATION of SHAPE {R=1 to 64; 0=Normal; 16=90' Clockwize}
PEEK (243).....FLASH MASK
PEEK (241).....SPEED (256 - current speed)
PEEK (234).....COLLISION COUNTER for shapes
PEEK (232-233).SHAPE TABLE starting address
POKE 231,S.....Sets SCALE of SHAPE
PEEK (230).....HI-RES PLOTING page. (32=Page 1, 64=Page 2, 96=Page 3)
PEEK (224-226).HI-RES GR X&Y Cordinates
POKE 228,x.....HI-RES GR COLOR BYTE (x can be 0-255)
```

HI-RES COLORS

```
0= Black1 {Gr/Vl}  1= Green          2= Violet     3= White1 {Gr/Vl}
4= Black2 {Or/Bl}  5= Orange         6= Blue             7= White2 {Or/Bl}
```

OTHER USEFULL CALLS
{Add +65536 to get pos. POKE's}

```
CALL 54915.....CLEARS STACK. Dose away with the false "OUT OF MEMORY" error.
CALL 1002......Reconnect DOS
CALL -8192.....RESET INTERGER BASIC. KILLS VARIABLES and CLEARS
CALL -8117.....LIST INTERGER PROGRAM
CALL -6739.....NEW
CALL -6729.....PLOTS a POINT on the screen
CALL -6090.....RUN INTERGER PROGRAM {SAVES VARIABLES}
CALL -4116.....RUN INTERGER PROGRAM {KILLS VARIABLES}
CALL -3973.....LOAD INTERGER PROGRAM from TAPE
CALL -3776.....SAVE INTERGER PROGRAM to TAPE
CALL -3774.....SAVE
CALL -3318.....CONTINUE
CALL -2458.....TURN ON MINI-ASSEMBLER
CALL -2423.....SWEET-16 INTERPRETER entry
CALL -1906.....MONITOR S/R DISASSEMBLER entry
CALL -1728.....MONITOR S/R-PRINT contents of X & Y {REG 9 as 4 HEX digits}
CALL -1716.....MONITOR S/R PRINT X BLANKS {X REG contains # to PRINT}
CALL -1402.....MONITOR S/R-IRQ HANDLER
CALL -1390.....MONITOR S/R-BREAK HNADLER
CALL -1370.....RE-BOOTS DISK SYSTEM
CALL -1321.....MONITOR S/R to display USER REGISTERS
CALL -1233.....MONITOR S/R SREEN INIT
CALL -1223.....MONITOR S/R set SCREEN to TEXT mode {Destroys ACCUMULATER}
CALL -1216.....MONITOR S/R set GRAPHICS mode {GR} {Destroys ACCUMULATER} CALL
CALL -1205.....MONITOR S/R set NORMAL WINDOW
CALL -1184.....Prints the 'Apple ][' at the top of your screen.
CALL -1181.....MONITOR S/R MULTIPLY ROUTINE
CALL -1148.....MONITOR S/R DIVIDE ROUTINE
CALL -1087.....MONITOR S/R CALCULATE TEXT BASE ADDRESS
CALL -1052.....MONITOR S/R SOUND BELL
CALL -1027.....MONITOR S/R OUTPUT A-REG as ASCII on TEXT SCREEN 1
CALL -856......MONITOR S/R WAIT LOOP
CALL -756......GET KEY from KEYBOARD {Destroys ACC & Y-REG} WAIT for KEY PRESS.
CALL -741......MONITOR S/R KEYIN ROUTINE
CALL -715......READ KEY & PERFORM ESCAPE FUNCTION if necessary.
CALL -678......Wait for RETURN
CALL -676......Bell; Wait or RETURN
CALL -670......PREFORM LINE CANCEL
CALL -665......PREFORM CARRIAGE RETURN & GET LINE of TEXT.
CALL -662......GET LINE of TEXT from KEYBOARD {X RETND with # of CHARACTERS}
CALL -657......INPUT; Accepts commas & collons.
             EX:PRINT "NAME (LAST, FIRST):";:CALL-657:A$="":FOR X= 512 TO 767
              IF PEEK (X) < > 141 THEN A$= A$ + CHR$ (PEEK (X) -128) : NEXT
CALL -626......PRINT CARRIAGE RETURN {Destroys ACCUMULATOR & Y-REG}
CALL -622......PRINT A1H,A1L. Example: 10 POKE 60,A1H  20 POKE 61,A1L   30END
            ...Then RUN, CALL -622
CALL -550......PRINT CONTENTS of ACCUMULATOR. As 2 HEX DIGETS.
CALL -541......PRINT a HEX digit
CALL -531......OUTPUT CHARACTER IN ACCUMULATOR. {Destroys ACCUM. & Y-REG COUNT}
CALL -528......GET MONITOR CHARACTER OUTPUT
CALL -468......PERFORM MEMORY MOVE A1-A2 TO A4.
             Example:    10 POKE 60,LOB
                     20 POKE 61,HOB
                     30 POKE 62,LOE
                     40 POKE 63,HIE
                     50 POKE 66,LOD
                     60 POKE 67,HID
            ...Then RUN, CALL -468
            * Note: LOB is lo-byte of begining of memory to move, HIB is

               high, LOE is low end, HIE is high, LOD is low destina-
                  tion, HID is high.
CALL -458......Perform MEMORY VERIFY (compare and list differences)
CALL -418......DISASSEMBLE 20 INSTRUCTIONS
CALL -415......DISASSEMBLER  Note: POKE start add. at 58-59 before calling.
CALL -378......set I FLAG
CALL -375......set KEYBOARD
CALL -336......JUMP to BASIC
CALL -333......CONTINUE BASIC
CALL -330......MEMORY LOCATION "GO"
CALL -321......DISPLAY A,S,Y,P,S REG. {CURRENT VALUES}
CALL -318......PERFORM MONITOR TRACE
CALL -307......WRITE OUT cassette tape
CALL -259......READ FROM cassette tape {LIMITS A1 to A2}
CALL -211......PRINT "ERR" & SOUNDS BELL {Destroys ACCUMULATOR & Y-REG}
CALL -198......PRINT BELL {Destroys ACCUMULATOR & Y-REG}
CALL -193......MONITOR & SWEET-16 "RESTORE"
CALL -188......MONITOR "RESTR1"
CALL -182......MONITOR & SWEET-16 "SAVE"
CALL -180......MONITOR "SAV1"
CALL -167......ENTER MONITOR RESET, TEXT mode, "COLD START"
CALL -155......ENTER MONITOR, ring BELL, "WARM START"
CALL -151......Go to MONITOR
CALL -144......SCAN INPUT BUFFER {ADDRESS $200...}
             EX: A$ = "300:A9 C1 20 ED FD 18 69 01 C9 DB D0 F6 60 300G D823G"
               FOR X=1 TO LEN(A$): POKE 511+X,ASC (MID$ (A$,X,1))+128: NEXT
               POKE 72,0: CALL -144
```

ERRORS

```
POKE 216,0.....RESETS ERROR FLAG
PEEK (216).....If = 127 then an ERROR was detected.
PEEK (212).....Returns ERROR CODE FLAG in decimal.
```

MEMORY ALLOCATION

RANGE	USE DESCRIPTION
$0-$1FF	Program wook space {not for USER}
$200-$2FF	Keyboard Character buffer
$300-$3FF	Available for short Machine langauge rutine
$400-$7FF	Screen display page 1 TEXT or GR
$800-$1FFF	Available RAM for BASIC programs
$2000-$3FFF	HGR page 1
$4000-$5FFF	HGR page 2
$6000-$95FF	Available RAM for BASIC programs
$9600-$9CFF	DOS files buffers {Maxfiles 3}
$9D00-$AAFC	Main DOS routines
$AAFD-$B7B4	File Manager
$B7B5-$BFFF	RWTS
$C000-$CFFF	I/O Hardware {end of RAM}
$D000-$FFFF	ROM {I/O Addresses}

SPECIAL MEMORY LOCATIONS

LOCATION	USE DESCRIPTION
$18	First track of data {for DOS}
$19	First sector of data {for DOS}
$1A	Number of sectors to load {for DOS}
$1B	The HIGH BYTE of the buffer {LO is always 00} {DOS Command}
$1A - $1B	Shape pointer used by DRAW and XDRAW
$1C	Last color used {HCOLOR converted to its color byte}
$26 - $27	Address of byte contained X,Y point
$2B	Boot SLOT * 16
$2C	Lo-res line END-point
$30	COLOR * 17
$33	Prompt-Char, {POKE 51,0:GOTO line #; Defeats NOT DIRECT COMMAND}
$68	LOMEM: {LOW BYTE is always 00}
$4E - $4F	Random - Number feild
$69 - $6A	Simple Variables
$6B - $6C	Start of ARRAY - Space

```
$6D - $6E  ! END of ARRAY - Space
$6F - $70  ! Start of STRING storage
$73 - $74  ! HIMEM: $73=LO BYTE
$75 - $76  ! Line # being executed
$77 - $78  ! Line # where program stopped
$79 - $7A  ! Address of executing line #
$7B - $7C  ! Current DATA line #
$7D - $7E  ! Next DATA address
$7F - $80  ! Input or Data address
$81 - $82  ! Last used Variable NAME: VAR$ = CHR$(PEEK(129)) + CHR$(PEEK(130))
$83 - $84  ! Last used variable address
$AF - $B0  ! End of Applesoft program
$D8       ! ONERR flag   NOTE: POKE 216,0 cancels ONERR function
$DA - $DB  ! Line # of ONERR error
$DE       ! ONERR error code {Dec. PEEK (222)}
$E0 - $E1  ! X-coordinate (0-279) in HEX {Low,High}
$E2       ! Y-coordinate (0-191) in HEX
$E4       ! Color being used {0=0:42=1:85=2:127=3:128=4:170=5:213=6:255=7}
$E6       ! Current HI-RES page being used {$20: Page one, $40: Page two}
$E7       ! Current SCALE (0-256)
$E8 - $E9  ! Location of shape table {Low,High}
$EA       ! Collision counter {used by XDRAW and DRAW}
$3D0 - $3D2! JUMP vector to DOS Warmstart {JMP $9DBF}
$3D3 - $3D5! JUMP vector to DOS Coldstart {JMP $9D84}
$3D6 - $3D8! JUMP vector to DOS File Manager {JMP $AAFD}
$3D9 - $3DB! JUMP vector to RWTS {JMP $B7B5}
$3DC - $3E2! Subroutine to locate File Manager PARM list {LDA $9D0F;LDY $9D0E}
$3E3 - $3E9! Subroutine to locate RWTS PARM list {LDA $AAC2; LDY $AAC1; RTS}
$3EA - $3EE! JUMP to replace DOS intercepts subroutine {JMP $A851; NOP; NOP}
$3EF - $3F1! JUMP vector to Autostart BRK Handler {JMP $FA59}
$3F2 - $3F3! Autostart Reset handler {$9DBF}
$3F4      ! POWER-UP byte ($3F3 EOR $A5) {$38}
$3F5 - $3F7! JUMP vector to Applesoft & Handler {JMP $FF58}
$3F8 - $3FA! JUMP vector to CTR-Y handler {JMP $FF65}
$3FB - $3FD! JUMP vector to NMI handler {JMP $FF65}
$3FE - $3FF! Vector for IRQ handler {$FF65}
$AA61.$AA60! LENGTH of file just loaded {$AA61 is the HIGH BYTE}
$AA73.$AA72! STARTING ADDRESS of file just loaded {$AA73 is the HIGH BYTE}
$FBB3     ! SIGNATURE byte {$06 = //e      :  $EA
 = ][+}
```

MISCELLANEOUS INFORMATION
CONTROL RESET

To make it run your program type this:

```
      10 POKE 1010,102
      20 POKE 1011,213
      30 POKE 1012,112
```

To make it send you to MONITOR type this:

```
      POKE 1010,105
      POKE 1011,255
      CALL -1169
```

To make it BOOT DOS type this:

```
      POKE 592,0
      POKE 1012,0
```

```
* Note: The origanal values are:
      PEEK(592) = 255  DivDos64k  Norml
      PEEK(1010)= 3       60     191
      PEEK(1011)= 224     191    157
      PEEK(1012)= 69      26     56
```

VERY QUICK SORTING ROUTINE

```
1000 FOR I = 1 TO N - 1 : REM N = # OF ITEMS
1010 P = I
1020 FOR J = I + 1 TO N
1030 IF A(J) < A(P) THEN P = J
1040 NEXT J
1050 T = A(I) : A(I) = A(P) : A(P) = T
1060 NEXT I
```

DOS MEMORY LOCATIONS

```
LOCATION    ! USE DESCRIPTION
-----------+-----------------------------------------------------------------+
$3D0 - $3D2 ! Re-enter DOS Vector
$3F2 - $3F4 ! Reset Vector   EX: POKE 1012,0 Reboots  {Norm: 56}
$3F5 - $3F7 ! Ampersand Vector. EX: POKE 1014,165:POKE 1015,214 -=> LIST
          !             EX: POKE 1014,110:POKE 1015,165 -=> CATALOG
          !             EX: POKE 1014,18 :POKE 1015,217 -=> RUN
$3F8 - $3FA ! Crtl - Y Vector
$A56E     ! Catalog Routine.  Also  CALL 42350
$9E42     ! Greeting program RUN-FLAG {POKE 40514,X: 52=BRUN, 20=EXEC}
$A884-$A907 ! DOS Commands
$A972-$AA3E ! ERROR messages
$A960-$AA61 ! Last BLOAD Lenght {LEN = PEEK (43616) + PEEK (43617) * 256}
$AA72-$AA73 ! Last BLOAD START {STR = PEEK (43634) + PEEK (43635) * 256}
$AA57     ! MAX Files Values
$AAB1     ! Max files Default Value
$AA68     ! Drive - Number  EX: POKE 43624,DR   DR= Drive for I/O
$AA6A     ! Slot - Number
$AC01     ! Catalog Track number.
$AE17     ! # Characters -1 in catalog file name.
$B3A7-$B3AE ! File type codes
$B3AF-$B3BA ! Disk Vol. Heading
$B3C1     ! Disk Vol. Number
$B3F0     ! Number of Sectors per Track
```

DOS MISCELLANEOUS

```
To defeat the "NOT DIRECT COMMAND" error type: POKE 51,0 : GOTO line #
To kill the INIT command do: Poke 42309,96  or  $A545:60
To kill the INIT command in normal DOS type: POKE 42309,96  or $A545:60
If you want a basic program to load in after HGR {more memory than LOMEM:16384}
   use this loader program:
       10 POKE 16384,0 : POKE 104,64 : REM STARTING LOCATION OF PROGRAM
       20 PRINT CHR$(4) "RUN PROGRAM"

  *Note:To put things back to normal use this program:
           10 POKE 2048,0 : POKE 104,8
           20 PRINT CHR$(4) "RUN OLD PROGRAM"

If PEEK(-18070) = 150 then your using DOS 3.3 .
```

```
POKE 40193,PEEK(40193)-N:CALL 42964...Move DOS buffers down N*256 bytes.
POKE 44452,N+1:POKE 44605,N...........Allows N file names before Catalog pause.
POKE 44505,234:POKE 44579,234:POKE 44580,234...Cancels return after file names.
POKE 44578,234:POKE 44579,234:POKE 44580,234...Cancels catalog pause.
POKE 44599,234:POKE 44600,234.........Wait for key input after every file name.

Here are some POKEs that turn on the DRIVES but do not READ or WRITE.  These
   can be used as scare tactics. EX. PRINT "INITIALIZING DISK" : POKE -16151,0
      POKE -16151,0........TURNS ON DRIVE 1
      POKE -16135,0........TURNS ON DRIVE 2
      POKE -16152,0........TURNS OFF DRIVE 1
      POKE -16136,0........TURNS OFF DRIVE 2

To stop CATALOG for a key input after every file name type:
      POKE 44599,234 {NORM 208}
      POKE 44600,234 {NORM 8}

To omit the pause after a full screen of CATALOG then type:
      $AE34:60    or  POKE 44569,96

For WILDCARD DOS files useing "=" type: (from monitor)
      B201:4C 71 BA
      BA69:E8 B1 42 DD C6 B4 D0 0A C8 C0 1E D0 F3
         AE 9C B3 18 60 C9 AD F0 F7 4C 0B B2
```

MISCELLANEOUS

```
To make the program in memory run when any Syntax (but DOS commands) is typed
   then put this line in:  10 POKE 214,128 {Norm 0}

PEEK (104).....If 8 is returned then APPLESOFT is in ROM. Any other value means
             APPLESOFT is in RAM or not available.
POKE 2049,1...Repeatedly Lists first  li
CALL -856......TIME DELAY. POKE 69,XX to set amount of delay.
CALL -1182.....Prints the Apple ][ across the top of your screen.

POKE 49107,234:POKE 49108,234:POKE 49109,234...Prevents language card re-load.

For "true" random number generation use RND(PEEK(78)+PEEK(79)*256).

POKE 1912+SLOT,1 on APPLE PARALLEL CARD (with P1-02 PROM) will enable LINEFEED.
POKE 1912+SLOT,0 on APPLE PARALLEL CARD (with P1-02 PROM) => disable LINEFEEDS.

REMAINDER {Mod} type: R = X - (INT (X / Y) * Y)
To ROUND to N digets past the decimal type: X = INT (X * (10^N) +.5) / (10^N)
QUADRATIC formula : R1 = (-B + SQR (B^2 - 4 * A * C)) / (2 * A)
                R2 = (-B - SQR (B^2 - 4 * A * C)) / (2 * A)
```

CONVERSIONS

```
To change VOLUME # xxx to SECTORS FREE = xxx then type the following:
      ADC0:20 69 BA
      BB69:A9 00 85 40 85 41 A0 C8 18 B9 F2 B3 F0 0E 0A 90 FB 48 E6 40 D0 02 E6
         41 68 18 90 F0 88 D0 E9 A6 40 A5 41 20 24 ED 60
      B3AF:A0 BD A0 D4 C3 C5 D3 A0 C5 C5 D2 C6

If you own a //e then you can get the functions of an 80 col card (save 80 col)
```

```
   with out the card. Just type: POKE 49162,0  then type: PR#3

[Dist. The Temple of Doom  805/682-5148]
+==============================================================================+
!  If you find an error or want to add something, please leave me a message!  !
+==============================================================================+
=============================================================================
DOCUMENT quick.draw.3
=============================================================================
```

,m3 0

```
                        ____________
                       /           /
                      / Section 9 /
                     /___________/
```

```
---------------
OTHER COMMANDS:
---------------
```

From the XTRAS menu, there are three additional commands:

RAM
The number of bytes of memory left is displayed.

NEW
Clears memory so that you can begin with a clean slate.

BOOT
The computer will be rebooted from the drive that Quick-Draw Adventure
Mapper was loaded from.

```
                        _____________
                       /            /
                      / Appendix I /
                     /____________/
```

```
-------------------------------
SPECIFYING INTERFACE CARD DATA:
-------------------------------
```

If you have a printer interface card that is not currently supported,
you have three options:

[01]
Buy a supported card. (Not a popular choice).

[02]
Write your own interface driver. (Explained in the next section).

[03]
Specify certain parameters about your interface. (Explained below).

During configuration, select the "USER-SPECIFIED" option for your
interface and answer affirmatively when asked if you want to specify
your interface parameters.

You need to know the following information which should be in the
manual for your interface card:

Data Address
This is the address at which each byte must be stored in order to transmit it to the printer.

Busy Address
This is the address that contains the printer busy status.

Busy Bit
This is the bit number in the busy address that must be tested to

determine if the printer is busy. The low order bit is 0 and and the high order bit is 7.

Set/Reset
Set means that if the bit is on, the printer is busy. Reset means that if the bit is off, the printer is busy.

Strobe On Address
This is an address that must be toggled after the data byte has been stored in order to transfer the data. It is not required on most interfaces.

Strobe Off Address
This address is required on some interfaces and must be toggled after the strobe on address.

Addresses can be entered as decimal (default) or hexadecimal (preceded with a "$") values. You can enter absolute addresses set for a specific slot or make them adjust to the slot number (s) entered during configuration. After specifying each address, you will have to select one of five address modifications:

- None
- Slot *16
- Slot *256
- 2nd Slot *16
- 2nd Slot *256

WRITING AN INTERFACE DRIVER:

If you can program in 6502 assembly language, you can write your own interface driver. The rules are very simple:

1. Your routine must begin at $4300 and be no longer than $0100 (256) bytes.

2. The printer drivers JSR to $4300 with the character to be output in the accumulator.

3. Name your routine "USER-WRITTEN.IF" and select "USER-WRITTEN" from the interface menu during configuration.

4. The printer slot # is stored at $CE. The 2nd slot value is stored at $CF.

5. You can use the page zero locations between $90 and $9F.

WRITING A PRINTER DRIVER:

Writing a printer driver is considerably harder. The following requirements should be met:

1. The driver must begin at $4000 and be no more than $300 (768) bytes long.

2. To be general purpose, it should call an interface driver at $4300 to output each character.

3. Mapper calls 5 different subroutines:

> JSR $4000 to perform a normal form feed operation on the printer.

> JSR $4003 to output a normal text line. The data begins at $2C00. Output 80 bytes maximum or until a carriage return ($D) is found.

> JSR $4006 to enter graphics mode. You can do whatever is required to init alize the printer. All subsequent calls will be to output graphic lines until a leave graphics mode call is made. The line spacing must be 7 "dots" high.

> JSR $4009 to send a line of graphics data. The data is at $2A00 and consists of 480 bytes with each byte containing a column that is 7 "dots" high and one "dot" wide. The high order bit is always off, the 2nd highest is the top bit in the column, the low order bit is the bottom.

> JSR $400C to leave graphics mode. You should do all of the "clean-up" that is required.

1. You can use the space between $2000 and $27BF while in graphics mode. The contents will be indeterminate when graphics mode is entered and will be destroyed after graphics mode is exited. In between, it will be stable and survive between graphic line calls.

2. Name your routine "USER-WRITTEN.PR" and select "USER-WRITTEN" from the printer selection menu during configuration.

3. You can use the page zero locations between $80 and $8F.

4. The graphics density value is stored at location $4B. A zero indicates single density, while a one means double density.

===
DOCUMENT quick.spells
===

FIRE: The Iron Spells

AMRAS	The Snake of Fire
ORLOS	The Flame of the Eye
AMRASMUR	The Copper Snake of Fire
ORLOSMUR	The Copper Flame of the Eye

The Silver Spells

AMRASAKIM	The Silver Snake of Fire
KUN	The Furnace of the Mind

The Gold Spells

AMRASLAZAR	The Golden Snake of Fire
KUNLAZAR	The Golden Furnace of the Mind

The Platinum Spells

AMRASTEL	The Platinum Snake of Fire
KUNTEL	The Platinum Furnace of the Mind

The Crystal Spell

CYQIEKUN	The Furnace of the Great Mystic

FROST: The Iron Spells

STRAL	The Frozen Hand
SEHK	The Breath of Ice

The Copper Spells

KRAMUR	The Tempest of Chaos
SEHKMUR	The Copper Breath of Ice

The Silver Spells

KRAAKIM	The Silver Gale of Chaos
SEHKAKIM	The Silver Breath of Ice

The Gold Spells

STRALLAZAR	The Golden Hand of Freezing
SEHKLAZAR	The Golden Breath of Ice

The Platinum Spells

KRATEL	The Platinum Maelstrom of Chaos
SEHKTEL	The Platinum Breath of Ice

The Crystal Spell

MUZAQ	Cacophonous Oblivion

PROTECTION: The Iron Spells

TEI	The Iron Shield
SEL	The Iron Armor

The Copper Spells

SELMUR	The Copper Armor
TASRAK	The Shell of the Unborn

The Silver Spells

SELAKIM	The Silver Armor
RESEN	The Globe of Peace
FSIRITH	The Talisman of Awe

The Gold Spells

TEILAZAR	The Golden Shield
AROMIR	The Water of Waking
SILAMEKSH	The Silken Shroud of Life

The Platinum Spells

RESNTEL	The Cyanic Globe of Peace
MUAMAAR	Phantasmal Terror

The Crystal Spell

QADIOS	The Womb of Infinate Safety
HEALING: The Iron Spell	
SHUM	The Simplest Salve
The Copper Spells	
SHUMMUR	The Copper Salve
HELAS	The Many Fingers of Healing
LUQMAR	Waking
The Silver Spells	
SHUMAKIN	The Silver Salve
ISO	Air
The Gold Spells	
SHUMLAZAR	The Golden Salve
HELASLAZAR	The Many Golden Fingers of Healing
ISUL	Cleaning Air
The Platinum Spells	
HELASTEL	The Many Platinum Fingers of Healing
SAMECLU	Purity of Sanctuary
The Crystal Spell	
CYQIETUR	Ceaseless Healing

KNOWLEDGE: The Iron Spells	
NGOS	The Glowing Script
LUM	Light
The Copper Spells	
NGOSMUR	The Glowing Copper Script
TALIS	The Spirit of Observtion
The Silver Spells	
ALTIS	The Psychic Key
EZAHM	The Water of Strength
The Gold Spells	
NGOSLAZAR	The Glowing Golden Script
LUMLAZAR	Golden Light
REAAM	True Revelation
The Platinum Spells	
LENTIIS	The Holding Heart
SATOR	The Mind Siphon
ROTAS	Suction
The Crystal Spell	
CYQIEQARAQK	The Tome of the Seer

==
DOCUMENT secretk.app
==

< Press SPACE to abort / ^S to Pause >

May 25, 1987

Hello again to everyone. Yes, it's that time again, time for another round of secret keys, time for "Hot Rod's Text File Number 3: Secret Keys //!" Much fanfare...

As always, I like to hear your questions, comments, and whatever else you may have to say. Any new finds are appreciated. You can currently reach me on the Curse BBS at (612) 926-5112. This number may change in summer 1987, so be prepared. It's a good board, we have a lot of fun, and it's the only place you can find me, unless you write a program and stick a secret key in it, then I'll find you.

Also note the new added section in this file called "VaporWares". If you have any info on these, please make an attempt to drop me a note. Thanks.

And now on to the secret keys (in keeping with tradition, they are in no particular order whatsoever. Really.)....

Intrigue - On the crack from the Bunnymen, type AMIGA at their title page for a secret screen.

Sea Dragon - Ctrl-J from the main page gives you lots of fuel.

Animate - A message is lurking on track 0, sector 9 (DOS 3.3). And if you pull down the "About" box and then press ? you get a nice note.

Fireworx - This is a packing program from Krackerjack, if you still have it. He tempted us with a secret key and it is H.LEWIS

Catsend - Ctrl-P gives a poem.

Mario Bros. - One the file crack from 202 Alliance, ESC at the page gives another page.

Flight Simulator][v1.0 - The one by MPG. Ctrl-A gives a message.

Newsroom - Ctrl-V gives the version number (during boot process.)

Space Ark - The MPG crack will give a secret message if you press shift-ctrl-]

Ghostbusters - You have a secret account! Name: Owen, Number: List. Gives scads of cash.

Print Shop Companion - I looked this one up since the last time. On the front side at the main menu, try STEVEN Esc to get a flip. On the back side, try Esc Ctrl-^ for a game (Driver).

Tass Times in Tone Town - During the intro music, press both buttons and Ctrl-B for a picture.

Infocom - Well, I thought I would mention this one. There is a secret subsystem in the later Infocom stuff. The commands are preceeded by a $, such as $ve, and $de. Those do a disk verify and debug respectively. They vary from game to game and release to release. There also is another set that begin with #, but I haven't figured them out. If any of you have a program called KIWI (Krell's Infocom Word Interpreter), it will print all the words out, including these commands. Then you can play with them. Some other things they do are to stop time, give you certain objects, take you to locations, etc. On some of the games, typing ZORK gets a humorous response. Oh well, just play around with them. And get KIWI.

Black Magic - This is a neat one. While going down a ladder and pressing button 0, press ^. There is a catch though, paddle 0 has to be at $8A (138), which is a little right of center as you hold it down. The game will beep when you get it right. Try calibrating the joystick before playing. Anyway, after that you can use ctrl-L for class, and ctrl-X to go to part 2 of the game. If you want it done permanently, look at track $05, sector $08, byte $29 for a 4C E7 1E. Try sector $0A if that's not right. Well, just change the 4C to a 2C.

Bug Attack - Ctrl-B during play will kill all the bugs for you. Use repeat.

Donkey Kong - I looked this one up too. During play, pressing 1 gives more men, 2 makes you invulnerable.

Jellyfish - " allows two player mode. Yes, that was a ".

Labyrinth - The one from Brody, by Scott Schram. I checked it out, Esc K A Y then a number 1-8 gives you that level of play. Watch this Schram guy!

Miner 2049er - At the place where it asks for the number of players, press # then 1-0 for the level you want.

Rearguard - Ctrl-T allows you to select level. Ctrl-L also does something, but I couldn't figure it out.

Serpentine - Esc ! $ during play gives you another serpent.

Snoggle - While you are dying, press ctrl-] to get 3 more men. And who says there isn't life after death?

Spare Change - Just in case you forgot, ctrl-Z allows you to do some editing.

Tubeway - Esc R $ allows you to select levels.

Choplifter - Ctrl-L followed by a number 1-3 gives you that level. Good ol Dan!

Lode Runner - Ctrl-^ goes to next level, ctrl-@ gives extra men.

Ultima IV - Ctrl-S will display a 16 digit number to be read as 8 two-digit numbers. They indicate how you are doing as far as becomming Avatar.

Airheart - Now supposedly typing LW at the title page gives you that Driver game, but I couldn't get it to work. I tried a couple different versions too. Anyone else get it?

Crime Wave - Scott Schram again! During play, press Esc MARK. Now you can do one of three things: A number 1-9 gives that level, Z gives permanent suicidal speed, or any other key gives 3 more guys up to 9.

I also checked Genetic Drift by Scott Schram and it does check for ctrl-^, but I couldn't see as it did anything. You may want to play with it.

Mystery Secret Key (what a bonus!)

I found this cryptic secret key being applied to Serpentine in an old text file, but it obviously doesn't apply (does it?). Maybe you can match the game it goes to:
  "ESC R T B 9 when over first mountain on level 2. (More bombs)."
I sure would like to know what game it is....

VaporWares

This is a list of old games that I have never seen, but have found references to in old magazines, etc. I have seen some ads for some of these, but never the games. As I collect these ancient relics, I really would appreciate it if anyone could come up with some of these and pass them up my way.

Ocean Knight - ?

Death Race '82 - From Avant Garde, written by Dave Jones.

Federation - From Avant Garde, written by Jim Haga.

Neutrons - Dakin5/Level10.

M.I.R.V. - Dakin5/Level10.

Invader Attack
Invader Roundup
Space Mouse
Space Scanner - All by Zeitgeist.

Moon Shuttle - By Datamost.

Sigma 7 - Bendelli softeware, 1983, written by Arganat.

Slime - Synapse. This probably never came out for Apple

Sword of Sheol - Winner's Circle. Same guys as Sanitron.

Arex - Adventure International.

Capture the Flag - Sirius. Probably wasn't released before they went under, but who knows?

Even if you can give me a description of any of these, I'd appreciate it. I have a pretty good collection of the really old stuff, if you are looking for anything. I have re-cracked a number of older things to regain original title pages and complete versions of things, but there still are a lot of programs that got butchered that need to be fixed. I'm also interested in hearing about those.

Until next time, write a game!

Hot Rod/Black Bag

Call the Curse (612) 926-5112.

```
[Volume 24][?/Help]:
(>
==============================================================================
DOCUMENT softkey
==============================================================================

MULTIPLAN ----  Only protected on tracks 0 thru 4 (on boot side of disk)
                1. Change 'End of Address' marks on those tracks to normal
                2. Change byte D on track 0, sector A, from CB to DE

ZORK I,II,III - INFOCOM (also works on most other infocom games)
INFIDEL,
STARCROSS,      1. Use Disun to make a backup copy (repair function 'on')
SUSPENDED,      2. Edit Trk $0, Sec $2, Byte 5D from BC to AD, Byte FB from
DEADLINE,          C9 to 29, Byte FC from BC to 00
PLANETFALL,
ENCHANTOR

DATA REPORTER - Protected on track 22, sectors 0 to 6
                1. Use Disun to make broken copy (repair function 'on')
                2. Run 'Hello' on original disk
                3. Choose 'Quit' option
                4. Replace original disk with broken copy
                5. Issue these commands: BSAVE HELLO.OBJ,A$9400,L$06FF
                                         LOCK HELLO.OBJ
                                         UNLOCK HELLO
                                         63999 PRINT D$;"BLOAD HELLO.OBJ":RETURN
                                         SAVE HELLO
                                         LOCK HELLO

PADDORA'S BOX - This will unlock most "Datamost" games
                1. Use Disun to make broken copy (repair function 'on')
                2. Change byte $91 on track 0, sector 3, from $DF to $DE
                3. Change byte $42 on track 0, sector 3, from $38 to $18

DONKEY KONG   - By "Atari"
                1. Use Disun to make broken copy (repair function 'on')
                2. Change the following bytes on the broken copy
                   Track 0, Sector 3: Byte $35 from $AA to $DE
                                      Byte $3F from $DE to $AA
                                      Byte $91 from $AA to $DE
                                      Byte $9B from $DE to $AA

CAVERNS OF    - By "MUSE" SOFTWARE
FREITAG         1. Use Disun to make broken copy (repair function 'on')
                2. Change abnormal Dos commands on broken disk
                3. Dos commands are on track 1, sectors 7 & 8
                4. Their 'Catalog' command is KSJFLKA (Change to 'Catalog') etc

APPLE LOGO ---- Fixed # of FF's between $D6's on track $1
                1. Use bit copier to copy tracks $0-$22 (error on trk $1 ok)
                2. Change the following bytes:
                   TRK    SCT   BYTE   FROM  TO
                   $00    $A    $13    $20   $EA       After making the required
                   $00    $A    $14    $00   $EA       changes, any bit copier
                   $00    $A    $15    $3D   $EA       can be used to make your
                   $00    $A    $22    $BD   $4C       duplicate copy.
                   $00    $A    $23    $8C   $55
                   $00    $A    $24    $C0   $40 ! * alternative bytes to 79-7B
```

```
                  * $00    $A    $79    $4C   $EA !   $5D from $4C to $EA
                  * $00    $A    $7A    $00   $EA !   $5E from $00 to $EA
                  * $00    $A    $7B    $C6   $EA !   $5F from $40 to $EA

ZAXXON -------- AN ARCADE FAVORITE
                1. Use Disun to make broken copy (repair function 'on')
                2. Change the following bytes on your broken copy

                                                  !  Non Mockboard versions,
                  For Mockingboard versions:      !  or Mockingboard versions
                                                  !  if other doesn't work
                   Trk    Sect   Byte   From   To ! Trk  Sect  Byte   From  To
                   $00    $04    $4F    $CC    $DE ! $00  $07   $00    $A9   $4C
                   $00    $04    $50    $D0    $EA ! $00  $07   $01    $01   $C0
                   $00    $04    $51    $AE    $EA ! $00  $07   $02    $48   $08
                   $00    $07    $0D    $A0    $4C ! $00  $04   $4F    $CC   $DE
                   $00    $07    $0E    $20    $D4 ! --------------------------
                   $00    $07    $0F    $84    $07 !
-------------------------------------------------- Trk  Sect  Byte   From  To
for super early >> $00    $07    $1F    $A9    $4C   $00  $04   $4F    $CC   $DE
versions           $00    $07    $20    $00    $C0   $00  $04   $50    $D0   $EA
                   $00    $07    $21    $85    $08   $00  $04   $51    $AE   $EA

                   To get more planes, change byte $17 on track $09 sector $08
                   from $02 to a greater number ($03 gives you 4, etc)

LEGACY OF ----- Breaking Sir Techs 'Window Wizardry' protection
LLYGAMYN        1. Use Disun to make a backup copy (repair function 'on')
                2. Change the following bytes on the boot side (side B)
                   starting at byte $15 on Trk $1A, Sect $0C
                   D0 16 EA AD 2D 00 CE FB 00 D0 F8 AD DE 00 A9 01
                   48 A5 01 48 A5 00 48 69 A9 00 F0 ED
                Edit Trk $6, Sect $A, byte $73, from $CB to $C3
                Remove serial # on Trk 0, Sect 5, bytes $01 thru $06,
                change all these bytes to $00's

                Also try this quickie, COPYA both sides of disk, Sect edit Trk
                $1A, Sec $D, Byte $AD, change $04 to $00, write prot boot side

THE ARTIST ---- Sierra On-Line
                1. Use Disun to make a backup copy (repair function 'on')
                2. Bload 'MAIN MENU' from your copy
                3. Enter monitor and make the following changes
                   4257:57, 4662:60   (4257 was 8A, 4662 was B9)
                4. Bsave MAIN MENU,A$4000,L$4D
                5. Lock 'MAIN MENU'

EDD 3 --------- A real nasty one
                1. Use a copy card to freeze processor, then enter monitor
                2. Change the following locations:
                   $113A from AD to A9, $113B from F1 to 0B
                   $113C from 1B to EA, $21D8 from 2C to 00
                   $21DF from 25 to 00, $21FE from 06 to 00
                3. Now use copy cards normal procedures for a 48K compressed
                   or noncompressed copy

DOLLARS & ----- 1. Make a back-up copy with a bit copier
SENSE           2. Change the following bytes on Trk $3, Sect $0, start at $8C
```

```
                   VERSION III.12
                   EA A5 02 38 E9 40 85 04 A5 03 E9
                   00 85 05 A0 3F B1 04 91 02 88 10 F9
                   For Version III.14, use Locksmith 5.0, default modes
RENDEZVOUS ---- Only side one of disk #1 is protected
WITH RAMA       1. Use Disun to copy side 1 of disk #1 (repair 'on')
                2. BLOAD IO
                3. Enter monitor, 1B5F:20 29 1C (was AD 82 C0) disables nib cnt
                4. BSAVE IO,A$A00,L$1512

BACK TO ------- Peachtrees General Accounting System
BASICS          1. Use COPYA on all 3 disks, then make the following changes
                   GENERAL LEDGER, Trk $13, Sct $0, Byte $AA
                                   change from $10 FB D9 to $4C 25 8D
                   ACCOUNTS RCVBL, Trk $12, Sct $C, Byte $A9
                                   change from $10 FB D9 to $4C 95 7F
                   ACCOUNTS PYBLE, Trk $13, Sct $8, Byte $A1
                                   change from $10 FB D9 to $4C D6 82

VISIDEX ------- By Visicorp
                1. Use COPYA to make a backup
                2. Delete the file named 'VISIDEX' on the backup
                3. Boot the copy, wait for 'FILE NOT FOUND' message
                4. BLOAD VISIDEX  from original disk
                5. Enter monitor, 60A3:69 FF, 6000G
                6. When drive stops, remove orig, insert copy, INIT VISIDEX
                7. Delete VISIDEX, BSAVE VISIDEX,A$803,L$4404

VISITERM ------ By Visicorp
                1. Use COPYA to make backup
                2. Change byte $DF on Trk $15, Sect $0E from $B0 to $90

MUSIC CONST --- By Electronic Arts
SET             1. Copy Trks $0-22 with a bit copier
                2. Boot a normal 3.3 disk
                3. LOAD A4 from copy
                4. Enter monitor, 913A:EA EA, BSAVE A4,A$4A00,L$4B60 (on copy)

SIERRA   ------ Software as listed
ON-LINE         1. COPYA the original disk
SOFTWARE        2. Use method #1, if that fails use method #2

                   METHOD #1
                    Program       Trk   Sec   Byte  From     To
                ---------------------------------------------------
                Screenwriter II   $1A   $0E   $00   $CE 03   $60 AD
                Version 2.2       $08   $0F   $00   $CE 03   $60 AD
                                  $0C   $0F   $00   $CE 03   $60 AD
                                  $17   $0F   $00   $CE 03   $60 AD
                ---------------------------------------------------
                The Dic*tion*ary  $10   $0D   $00   $CE 03   $60 AD
                ---------------------------------------------------
                Sammy Lightfoot   $05   $0E   $00   $CE 03   $60 AD
                (or also try)     $0D   $00   $9B-9D  to   EA EA EA
                ---------------------------------------------------
                Time Zone         $03   $0F   $00   $CE 03   $60 AD
                Version 1.1
                ---------------------------------------------------
                Apple Cider       $12   $01   $00   $CE 03   $60 AD
                Spider
                ---------------------------------------------------
                Oil's Well        $10   $0F   $00   $CE 03   $60 AD
                ---------------------------------------------------
SIERRA          Program           Trk   Sec   Byte  From     To
ON-LINE         ---------------------------------------------------
CONT.           Cannonball        $18   $06   $00   $CE 03   $60 AD
                Blitz
                ---------------------------------------------------

                   METHOD #2   (Try this if a program seems to work but hangs)
                1. Search the disk for a byte sequence of $CE 03
                2. Look at the byte which lies eight bytes past the $C3
                3. If the byte is $09 (for example), then search the disk
                   for a JSR $0900 (20 00 09)
                4. Change the JSR to EA EA EA
                ----------------------------------------------------------
                Sammy     Trk C, Sect 3, Byte 69-6B & 73-75 to EA EA EA
                Lightfoot Trk 10, Sect B, Byte 81-83 &8B-8D to EA EA EA
                -----------------------------------------------------------
                Time Zone         Trk   Sec   Byte  From        To
                Version 1.1       $03   $0B   $F0   $20 00 17   $EA EA EA
                -----------------------------------------------------------

THE REPORT      By Sensible Software
CARD            1. Use Disun to make a backup copy (repair function 'on')
                2. Put your favorite fast loading dos on this backup
                3. Boot program should be 'HELLO'

SARGON III      Excellent Chess game
                1. Use Disun to make a backup copy (repair function 'on'
                2. Edit Trk $0, Sec $0, Byte $XX, change to
                3. Edit Trk $0, Sec $3, Byte $xx, change to

WIZARDRY I ---- The Proving Grounds of The Mad Overlord, by Sir Tech
                1. Use COPYA to copy both sides of the disk
Version         2. Edit Trk $22, Sec $04, starting at byte $A4 enter the
dated              following code on the boot side of your copy:
10-Mar-82          A0 00 AE 28 8B BD 29 8B 85 0D 91 02 E8 C8 BD 29
                   8B 91 02 85 0E E8 8E 28 8B 60 00 10 12 57 05 13
                   12 8C 05 08 12 53 05 E7 29 8D 05 17 12 53 05 12
                   12 92 05 D0  (write the sector back to the disk)
                   Put a write protect tab on this side
                3. Edit Trk $13, Sec $00, starting at byte $C0 enter the
                   following code on the 'Scenario Master' side of your copy
                   AE DC 20 BD DD 20 85 0D 8D 00 20 E8 BD DD 20 8D
                   01 20 85 0E E8 8E DC 20 8D E8 C0 60 00 23 12 61
                   06 23 12 62 06  (write the sector back to the disk)

WIZARDRY II     Knight of Diamonds, by Sir Tech
                1. Use COPYA to copy both sides of the disk
Version         2. Edit Trk $22, Sec $0E, starting at byte $CB enter the
dated              following code on the boot side of your copy
10-Mar-82          A0 00 AE 7F 8C BD 80 8C 85 0D 91 02 E8 C8 BD 80
                   8C 91 02 85 0E E8 8E 7F 8C 60 00 C9 09 42 09 57
                   09 2E 05 C7 09 4B 09 4B 09 26 05
                   write the sector back to the disk, and put a write protect
                   tab on this side
                3. Edit Trk $12, Sec $01, starting at byte $C0 enter the
                   following code on the 'Scenario Master' side of the copy
                   AE D9 20 BD DA 20 8D 00 20 85 0D E8 BD DA 20 E8
                   8D 01 20 85 0E 8E D9 20 60 00 8D 09 20 05 27 05
                   81 05 B7 E0 00  (write the sector back to the disk)
WIZARDRY BACK UP --- COPYA, Then Locksmith Trk A-E SYNC with parameter changes
                     46=96, 21=02

ULTIMA II       Sierra On-Line
                1. Boot your 3.3 system master
                2. Enter monitor, AFF7G  (Allows reading VTOC into memory)
                3. After drive stops type, AFF7:60, AFFD:60  (prevents DOS from
                   writing or reading altered VTOC from ultima disk
                4. Run COPYA on all 3 Ultima II disks (Also any character disk)
                5. Boot 3.3 system master again, leave disk in drive
                6. Enter monitor, 300:20 F7 AF 20 0C FD 20 FD AF 60
                7. Type 300G; when cursor reappears remove 3.3 syst master
                8. Place first Ultima disk (copy) into the drive, press any key
                9. Place next Ultima disk into drive, type 303G, also do this
                   for the last Ultima disk
               10. Insert the Ultima II Program Master disk in the drive and
                   type 'BLOAD HELLO', then enter the following changes:
                   72E0:A9 4C 8D F8 03 A9 79 8D F9 03 A9 50 8D FA 03 60
               11. Type 'UNLOCK HELLO', BSAVE HELLO,A$6000,L$1420
               12. Type LOCK HELLO

                ALTERNATIVE SOFTKEY TO ULTIMA II
                1. COPYA all 3 disks
                2. Sector Edit Ultima II Program Master, Trk $11, Sec $00
                   Byte $01 from $FF to $11, Byte $02 from $FF to $0F
                3. Sector Edit Program Master, Trk $3, Sec $C, Bytes $84,85,86
                   from $20 E0 72 to $EA EA EA
                4. Perform step #2 on copies of Player Master & Galactic disks

WITNESS,    --- Infocom Inc.
DEADLINE,       1. Use Disun to make a backup copy (repair function 'on')
STARCROSS       2. Edit Trk $0, Sec $2, change the following bytes:
                   $5D from $BC to $AD
                   $FB from $C9 to $29
                   $FC from $BC to $00 (also try $AD)

PRISONER II     Eduware  (game uses trk 35 for copy protection)
                1. Use COPYA to make a backup copy (track 35 not needed)
                2. UNLOCK IF.SHAPE
                3. BLOAD IF.SHAPE
                4. Enter monitor, 57B4:BD 8C  (old values are FE 57)
                5. BSAVE IF.SHAPE,A$5600,L$026E
                6. LOCK IF.SHAPE

PEST PATROL --- Sierra On-Line
                1. Cold boot with no disk in drive, hit reset, enter monitor
                2. Type 800:00, then type 801<800.BFFFM  (zero's mem 800-BFFF)
                3. Type 9600<C600.C6F7M (moves bootcode)
                4. Type 9600G, type the following while drive is running
                   86F:00, 801G, B8A4:00, B8A7:00, B800G, B375:00, B2E0G,
                   B47AG, B466:00, B4BEG, Then type 805:A9 00 8D F2 03 A9 E0 8D
                   F3 03 49 A5 8D F4 03 D0 0D, 8DC:4C 00 40
```

```
                 5. Type 9600<800.8FFM
                 6. Insert a 48K slave with no hello program, Type C600G
                 7. Enter monitor, Type 800<9600.96FFM
                 8. Insert a blank initialized disk
                    Type BSAVE PEST PATROL,A$800,L$7FFF
                 9. Use Copy II+ to change boot program to PEST PATROL

AZTEC --------- By Datamost
                 1. Use Disun to make a backup copy (repair function 'on')
                 2. Edit Trk $0, Sec $3, Byte $42 from $38 to $18

VISIFILE ------ This is an easy one
                 1. Copy the original disk with any copyer
                 2. Change byte $2D on track $22, sector $4, from $0A to $0F

DB MASTER   --- Stoneware, Inc.  (New version 'ProDOS' is not protected)
old version      1. Load COPYA, and add the following lines
                    199 GOSUB 400
                    248 GOSUB 420
                    259 GOSUB 420
                    400 POKE 47413,223:POKE 47423,171:POKE 47505,223:POKE
                        47515,171
                    405 POKE 48351,201:POKE 48352,12:POKE 48353,105:
                        POKE 48354,0:POKE 48355,24:POKE 48356,76:POKE 48357,107:
                        POKE 48358,190
                    410 POKE 48741,223:POKE 48742,188: RETURN
                    420 POKE 48741,107:POKE 48742,190:POKE 47413,222:POKE 47423,
                        170:POKE 47505,222:POKE 47515,170
                    425 POKE 48741,107:POKE 48742,190: RETURN
                 2. SAVE COPYA DB
                 3. Use COPYA DB to make the backup copy
                 4. Sector edit the copy and make these changes
                    Trk   Sct   Byte    From    To  !   Protection schemes used
                    ------------------------------!----------------------------
                    0     3     $35     DF      DE  ! Program uses 1/2 tracks from
                    0     3     $3F     AB      AA  ! $6.5-$22.5
                    0     3     $91     DF      DE  !
                    0     3     $9B     AB      AA  ! Closing addr & data marks
                    0     E     $0A     A2      D0  ! changed from DE-AA To DF-AA
                    0     E     $0B     00      12  !
                    1     F     $C7     A9      60  ! There is a nibble checking
                    3     1     $3E     20      60  ! routine to check trk 0

MASK OF THE --- By Ultrasoft Inc.
    SUN          1. Use Disun to copy both sides of the original disk with the
                    repair function 'on'
                 2. Delete the file 'LL(V27)' on side one of the copy
                 3. Use Copy II+ to change the hello program to 'DISK'

HOMEWORD ------ By Sierra On-Line
                 1. Use COPYA to make a backup
                 2. Edit Trk $10, Sec $0A, Bytes $9-A from 49 C9 to EA 60

DARK CRYSTAL -- By Sierra On-Line
                 1. Use COPYA on all 4 sides (only disk #1, side A is protected)
                 2. Sector Edit Trk $5, Sec $F, change
                    Bytes $A8-$AA from 20 F0 5F to EA EA EA
                 3. Edit Trk $7, Sec $C, change Bytes $22-$24
```

```
                      from 20 F0 F5 to EA EA EA

LANCASTER ----- Electronic Arts
Original        1. Use Disun to make a backup (repair function 'on')
version only    2. Change boot program to 'LANCASTER'

                Above procedure does not work on any later versions

VISIFILE ------ Visicorp
                1. Load COPYA, change line 250 as follows
                   250 PRINT "INIT XXX,S" SS ",D" SD",V" PEEK(714)-1:FT=1
                2. Run modified COPYA on both Visifile disks
                3. Edit disk #1 only, Trk $22, Sec $04, Byte $2D from 0A to 0F

SCREENWRITER    Sierra On-Line
    II          1. Use COPYA to make a backup copy (then hide the original)
version 2.2     2. Enter monitor, BLOAD RPART1, 1F90:EA EA EA
                3. BSAVE RPART1,A$C00,L$1400
                4. BLOAD EDITOR PART2.OBJ0, 1F49:EA EA EA
                5. BSAVE EDITOR PART1.OBJ0,A$C00,L$1400

                Update..Edit Trk $0E, Sec $03, locate sequence 20 00 6E, and
                change it to EA EA EA. Edit Trk $0F, Sec $07, locate 20 00 7F
                and change it to EA EA EA

HOME       ---- Continental Software
ACCOUNTANT      1. Use Disun to make a backup copy (repair function 'on')
                2. Add a custom DOS to make loading and accessing faster

CANYON CLIMBER  Same as Home Accountant

PANDORA'S BOX - Datamost Inc.
                1. Use Disun to make a backup copy (repair function 'on')
                2. Edit Trk $0, Sec $3, change byte $42 from $38 to $18,
                   change byte $91 from $DF to $DE

DONKEY KONG --- Atari Inc.
                1. Use Disun to make a backup copy (repair function 'on')
                2. Edit Trk $0, Sec $3, change byte $35 from $AA to $DE,
                   byte $3F from $DE to $AA, byte $91 from $AA to $DE,
                   byte $9B from $DE to $AA

LOCKSMITH ----- Omega Microware
   5.0          1. Edit Trk $F, Sec $E, Byte $6F from $C6 to $0F
                2. Use any copier

ROBOTRON ------ An easy one
                1. Use Disun to make a backup copy (repair function 'on')
                2. Delete the file 'Runner'
                3. Use Copy II+ to change the boot program to 'ROBOTRON'
```

```
==============================================================================
DOCUMENT trace2.app
==============================================================================

             THIS PHILE WAS DONATED BY MR. MADNESS
                  SYSOP OF THE
             <<<<<<<<< S H I R E >>>>>>>>

            ***************************************
            *                            *
            *       MR. XEROX'S BOOT TRACING      *
            *          PART  I               *
            *                            *
            ***************************************
```

NOTE: I CHOSE APPLE GALAXIN HERE BECAUS E IT IS A WIDELY DISTRIBUTED PROGRA M
, AND IT ENCOMPASSES THE BASIC ID E AS IN BOOT TRACE CRACKING.

FOR ALL THOSE INTRESTED PIRATES OU T THERE, YES THERE IS ANOTHER WAY TO CRA C K
PROGRAMS. YOU DON'T NEED ANY RAM-CAR DS,PROM BURNERS, OR FOREIGN TO REGULAR D
O S PROGRAMS, ANYBODY WHO IS NOT A CLOWN, WITH SOME MACHINE LANGUAGE PROGRAMMIN
G ABILITY CAN TRACE A BOOT. THIS METHOD OF CRACKING, TRACEING THE BOOT, IS IN
A

TRUE SENSE, CRACKING THE CODE.   YOU SEE, FOR ALL DISKS, THEY MUST FIRST BOOT
UP
T O START RUNNING. AFTER THE FIRST STAGE BOOT (AT LOCATION $C600), THEY JUMP
TO

SECOND STAGE BOOT PROGRAM (AT $800), AN D THEN TO A THIRD, AND SOME EVEN A
FORTH , BUT THERE COMES A POINT WHERE THE LOAD ING OF THE PROGRAM FROM DISK
STOPS, AND T HE RUNNING OF THE PROGRAM BEGINS. IF Y OU CAN TRACE THIS, AND
STOP IT AFTER IT I S FINISHED LOADING, AND SAVE ALL THE ME MORY LOCATIONS THAT
CONTAIN THE PROGRAM O NTO A NORMAL 3.3 DISK, YOU HAVE CRACKED THE PROGRAM.
THIS METHOD IS MOST USEFU L FOR CRACKING THE "SINGLE-SHOT" BOOTING PROGRAMS
SUCH AS APPLE PANIC, RASTER B L ASTER, AND GORGON. THESE DISKS DON'T CO NTAIN
ANY STANDARD DOS, BUT RATHER THEIR

OWN. THIS DOS HAS JUST ONE PURPOSE, AND THAT IS TO LOAD THE PROGRAM INTO THE
CO M PUTER, FROM THE DISK, AND START ITS EXE CUTION. NOW, THIS IS NOT AS
SIMPLE AS I T SOUNDS, AS THE SOFTWARE PROTECTORS ARE NOT DUMB, THEY TRY TO MAKE
IT TOUGH FOR

YOU TO TRACE. HOWEVER, IT IS NOT IMPOS SIBLE, SINCE THE DISK MUST BOOT UP, AND
S INCE IT MUST HAVE SOME BOOTING PROCESS, THAT IS TRACEABLE. LET ME TRY AND
SHOW YOU AN EXAMPLE OF HOW TO TRACE A BOOT OF A PROGRAM.LET

ME SHOW YOU HOW TO TRACE APPLE GALAXIAN . THE FIRST STAGE BOOT STARTS AT
$C600.

IF YOU TURN YOUR APPLE ON, AND TYPE " CALL-151 (RETURN)" AND "C600G (RETURN)",

THE DISK WILL PROCEED TO START AND BOOT THE DISK IN THE DRIVE.   THIS IS BECAUSE
$ C600 CONTAINING THE PROGRAM FOR THE DIS K TO BOOT FIRST. IF, YOU EXAMINE
THIS P R OGRAM BY TYPING "CALL-151 (RETURN)", AN D "C600LLLLLLL (RETURN)", YOU
WILL SOON C OME ACROSS A JMP $801, NEAR THE END, SP ECIFICALLY, AT $C6F8. THIS
IS THE LINK T O THE NEXT STAGE OF THE BOOT WHAT WE MUS T DO IS ALLOW THE FIRST

STAGE TO LOAD IN

AT $800, BUT INSTEAD OF LETTING IT RUN (CONTINUE TO BOOT, AND GO TO $800), STOP

THE COMPUTER, AND EXAMINE WHAT IS AT $8 00. TO DO THIS LETS MOVE $C600 DOWN TO $ 9600.TYPE "CALL-151 (RETURN)" AND "9600 <C600.C700M (RETURN)" THIS MOVES C600 DO W N FOR YOU. THEN TYPE"96F8:4C 59 FF (RET URN)", THIS WILL, INSTEAD OF HAVING THE B OOT GOTO $800, WILL MAKE IT JUMP TO $FF 59 (THE RESET LOCATION). THEN TYPE "9600 G ". YOUR DISK SHOULD BOOT UP FOR A SECO ND OR SO, AND THEN YOU SHOULD HEAR BELL,

AND THE MONITOR CURSOR WILL APPEAR AT T HE BOTTOM OF THE SCREEN.THE NEXT STEP IS

TO EXAMINE THE BOOT AT LOCATION $800. I F YOU LOOK AT THIS BY TYPING "800L (RETU R N)" YOU WILL SEE THE SECOND STAGE BOOT OF APPLE GALAXIAN. BY TYPING "800LLLLLLL

(RETURN)", YOU CAN SEE WHAT GOES ON NEX T IN THE BOOT STEP. WHAT HAPPENS NEXT, I S THAT IT TAKES THE MEMORY THAT IS STORE D AT $800, AND MOVES IT DOWN TO $200, AN D SOME OTHER STUFF, LIKE LOADING THE NEX T STAGE OF THE BOOT, AND THEN, IF YOU LO O K AT LOCATION $841, YOU WILL SEE A JUMP TO $301. THIS IS THE NEXT STAGE IN THE B OOT. SO, WE MUST MOVE WHAT IS IN MEMORY UP, OUT OF $800, BECAUSE THE NEXT TIME W E BOOT THE DISK, THE LOCATIONS AT $800 WILL BE CHANGED, SO TYPE "9800<800.900M (RETURN)", AND THAT WILL DO THE MOVE. TH E NEXT THING TO DO, IS TO CHANGE WHAT IS

AT $9800, THE STUFF WE JUST MOVED UP, S O THAT IT WILL RUN AT $9800, INSTEAD OF I TS NORMAL LOCATION OF $800. TO DO THIS, TYPE " 9803:BD 0 98 (RETURN)" AND "9841 : 4C 01 93 (RETURN)". THEN TYPE "9301:4C 59 FF", BECAUSE WE CHANGED IT TO RUN AT $ 9800, AND ALSO CHANGED IT TO STOP AFTER DOING THIS INSTEAD OF JUMPING TO THE NE X T BOOT STAGE, AT $300. WE TOLD IT TO JU MP TO $9300, AND AT $9300, WE PUT A JMP $ FF59 (JUMP TO RESET). AND FINALLY, CHAN GE THE JMP AT $96F8 FROM $FF59 TO $9801 B Y TYPING "96F8:4C 01 98". NOW AGAIN TYP E $9600G. THIS TIME, WE ARE ONE STAGE FARTHE R, IF YOU NOW MOVE THE STUFF AT $300 UP T O $9300, AND CHANGE IT TO WORK AT $9300 BY TYPING "9300<300.400M (RETURN)" AND " 9313:AD CC 93 (RETURN), AND "933C:AD CC 93 (RETURN)", THIS WILL BE COMPLETED. B U T NOW, THERE IS A PROBLEM.   THE JUMP OUT IS AT $9343, AND IT JUMPS NOT TO THE NE X T STAGE IMMEDIATELY, BUT TO A CERTAIN A MOUNT OF SUBROUTINES, AND AFTER THEM , T H ROUGH THE SAME JUMP, JUMPS TO THE NEXT STAGE.   HOW DO WE GET AROUND THAT YOU ASK

? THE ANSWER IS TO WRITE A PROGRAM THAT CHECKS TO SEE WHERE IT IT JUMPING TO, A N D IF IT IS NOT JUMPING TO WHERE IT NORM ALLY JUMPS TO, THEN STOP, BECAUSE WE KNO W THAT THE NEXT JUMP IS NOT TO A SUBROUT INE, BUT TO THE NEXT STAGE OF THE BOOT. T HIS MAY SOUND COMPLICATED, BUT JUST TYP E THIS ROUTINE IN AT $9400, "9400:A5 3E C 9 5D D0 03 6C 3E 00 4C 59 FF", AND "934 3:4C 00 94 (RETURN)". THAT WILL TAKE CAR E OF THIS STAGE. NOW CHECK TO SEE THAT Y OU HAVE TYPED IN EVERYTHING CORRECTLY, A N D THEN TYPE "9600G", TO RESTART THE BOO T. NOW, THE DISK SPINS FOR A LITTLE W HILE LONGER, AND THEN IT STOPS, WE HAVE C OME TO THE LAST STEP OF THIS BOOT PROCE SS. THIS STEP LOADS THE PROGRAM IN FROM D ISK, AND THEN JUMPS TO THE BEGINNING OF IT .BY TYPING "93CC (RETURN)", THE COMP U TER WILL DISPLAY THE PAGE-1 OF THE NEXT STAGE BOOT.   IT WILL DISPLAY "B6", AND Y O U ADD ONE TO IT, AND GET $B7, SO TYPE " B700L". AND PRESTO, WE HAVE THE NEXT STA G E OF THIS BOOT. THIS BOOT FROM HERE DOE S THE PROGRAM LOADING, ALONG WITH TURNIN G ON THE GRAPHICS, AND JUMPS TO THE BEGI NNING OF IT. IF YOU CAN SEE IT, THE BEGI N NING OF IT IS AT $600, AND THERE IS A J UMP TO $600 AT LOCATION $B759. SO, ALL W E HAVE TO DO IS TO HAVE IT DO ALL THE LO

ADING, AND INSTEAD OF HAVING IT JUMP TO $ 600, STOP IT THERE. BUT THERE IS A PROB LEM CONNECTED WITH THIS (ARN'T THERE ALW A YS !). THE PROBLEM IS THAT IF WE STOP I T HERE, LOCATION $600 IS IN TEXT VIDEO M E MORY, SO WE MUST NOT HAVE IT JUMP TO $F F59 (STOP), BUT JUMP TO A ROUTINE THAT R E LOCATES EVERYTHING FROM $0000-$0800, AN D THEN STOP. I WILL PROVIDE YOU WITH THI S . JUST TYPE "B500:A2 00 B5 00 9D 00 20 BD 00 01 9D 00 21 BD 00 02 9D 00 22 BD 0 0 03 9D 00 23 BD 00 04 9D 00 24 BD 00 05 9D 00 25 BD 00 06 9D 00 26 BD 00 07 9D 0 0 27 E8 D0 CE 4C 59 FF (RETURN)" THIS W ILL TAKE CARE OF MOVEING EVERYTHING FROM $0-$800 TO $2000-$2800. BUT NOW CHANGE $B759 TO JUMP TO THIS SMALL PROGRAM BY T Y PING "B759:4C 00 B5" BUT WE ALSO HAVE T O CHANGE SOME OTHER LOCATIONS. LOCATION $ 93CC MUST BE CHANGED TO $D6, SO TYPE "9 3CC:D6 (RETURN), AND INSTEAD OF JUMPING T O $FF59 AT $8409, AND STOPPING AT THAT STAGE OF THE BOOT, JUMP TO THE BEGINNING

OF THIS BOOT AT $B700, BY TYPING "9409: 4C 00 B7 (RETURN)". THAT TAKES CARE OF M O ST ALL PREPERATIONS FOR THE FINAL CRACK . NOW CHECK TO SEE THAT YOU HAVE TYPED I N EVERYTHING CORRECTLY, AND IF YOU ARE R EADY, TYPE "9600G" IF EVERYTHING WORKED CORRECTLY, IT SHOULD BOOT UP FOR ABOUT 10 SECONDS, AN D YOU SHOULD SEE THE HI-RES PICTURE LOAD ING IN, AND THEN YOUR SPEAKER SHOULD BEE P , AND YOU SHOULD SEE, ON THE SCREEN A B UNCH OF LETTERS.   IF THIS DIDN'T HAPPEN, C HECK ALL THESE STEPS, AND REPEAT THE PR OCESS. IF IT HAS, THEN YOU ARE JUST ABOU T FINISHED. IF YOU WANT TO CHECK TO SEE IF IT HAS WORKED, ASSEMBLE THIS PROGRAM,

AND TYPE IT IN AT $B560, IF NOT, GO ON TO THE NEXT STEP.

OBJ $B560 BEGIN LDX #$00 AGAIN LDA $2000,X STA $00,X LDA $2100,X STA $100,X LDA $2200,X STA $200,X LDA $2300,X STA $300,X LDA $2400,X STA $400,X LDA $2500,X STA $500,X LDA $2600,X STA $600,X LDA $2700,X STA $700,X INX BNE AGAIN ;LOOP JMP $0600 ;BEGINNING OF PGM NO W BOOT UP A NORMAL DOS DISK, AND SAVE EVE RYTHING FROM $2000-$2800, WHICH REPRESEN T LOCATIONS $0-$8 MOVED UP BY $2000.YOU SHOULD THEN REPEAT THE WHOLE BOOT TRACE,

AND PROCEED TO THE NEXT STEP.EXAMINE TH E MEMORY OF YOU APPLE, YOU WILL SHOULD S A VE ALL THE INFORMATION FROM $800-$A000 ON A NORMAL DOS DISK, THEN LINK THE FILE S THAT YOU HAVE SAVED ON THE DOS DISK TO GATHER, AND MAKE THE FILE A B-RUNABLE FI L E, THAT LOADS EVERYTHING IN, AND MOVES THE $00-$800 IMAGE BACK DOWN IN MEMORY,

AND THEN JUMPS TO LOCATION $600, THE BE GINNING OF THE PROGRAM.

IF YOU HAVE ANY QUESTIONS ON THIS, YOU MAY MAIL THEM TO ME. ALSO, I HAVE R E CENTLY CRACKED MANY GOOD PROGRAMS SUCH AS STAR BLAZER, TWERPS, SNAKE BYTE, GUAR D IAN, FOOSBALL, DUNG BEETLES, AND LOCKSM ITH 4.1. IF YOU ARE IN NEED OF ANY OF TH E SE, LEAVE ME MAIL ON THIS BOARD. LOOK F OR SOME NEW ARTICALS SOON, ON HOW TO CRA C K OTHER PROGRAMS, AND UNTIL THEN KEEP O N CRACKING ! IF ANY ONE OF YOU ARE UNFAMILIAR WITH H OW TO SAVE EVERYTHING, AND YOU NEED SOME

HELP, HERE IS HOW TO DO IT: FOLLOW THE DIRECTIONS FOR TRACEING THE BOOT, AND TYPE "2800<9600.A000M (RETUR N)" AND "3200<800.900M (RETURN)" ALSO, W E NEED A PROGRAM TO MOVE EVERYTHING THAT

WE JUST RELOCATED BACK INTO THEIR ORIGI NAL LOCATIONS.   SO WE NEED A PROGRAM LIKE

THIS:
```
      ORG $3400
      LDX #$00
```

```
LOOP1 LDA $2000,X
      STA $00,X
      LDA $2100,X
      STA $100,X
      LDA $2200,X
      STA $200,X
      LDA $2300,X
      STA $300,X
      LDA $2400,X
      STA $400,X
      LDA $2500,X
      STA $500,X
      LDA $2600,X
      STA $600,X
      LDA $2700,X
      STA $700,X
      NOP
      LDA $3200,X
      STA $800,X
      LDA $3300,X
      STA $900,X
      NOP
      LDA $2800,X
      STA $9600,X
      LDA $2900,X
      STA $9700,X
      LDA $2A00,X
      STA $9800,X
      LDA $2B00,X
      STA $9900,X
      LDA $2C00,X
      STA $9A00,X
      LDA $2D00,X
      STA $9B00,X
      LDA $2E00,X
      STA $9C00,X
      LDA $2F00,X
      STA $9D00,X
      LDA $3000,X
      STA $9E00,X
      LDA $3100,X
      STA $9F00,X
      NOP
      INX
      BNE LOOP1
      LDA $C057
      LDA $C054
      LDA $C052
      LDA $C050    ;GRAPHICS
      JMP $600       ;BGN OF PGM.
```

THIS TIME, I WILL ASSEMBLE IT FOR YOU, ALL YOU HAVE TO DO IS TYPE "3400:A2 0 BD 00 20 95 00 BD 00 21 9D 00 01 BD 00 22 9D 00 02 BD 00 23 9D 0 03 BD 00 24 9D 0 4 BD 0 25 9D 0 5 BD 0 26 9D 0 6 BD 0 27 9D 0 7 EA (RETURN)" AND "3432:BD 0 32 9D 0 8 BD 0 33 9D 0 9 EA (RETURN)" AND "34 3F:BD 0 28 9D 0 96 BD 0 29 9D 0 97 BD 0 2 A 9D 0 98 BD 0 2B 9D 0 99 BD 00 2C 9D 0 9A BD 0 2D 9D 0 9B BD 0 2E 9D 0 9C BD 0

```
2F 9D 0 9D BD 0 30 9D 0 9E BD 0 31 9D 0 9F (RETURN)" AND "347B:E8 D0 84 EA AD 5
7 C0 AD 54 C0 AD 52 C0 AD 50 C0 EA 4C 00 06 (RETURN)".  THIS WILL TAKE CARE OF
TH E SMALL PROGRAM THAT WE NEED TO MOVE EVE RTHING BACK.  BUT WE ALSO NEED TO
PUT A J M P $3400 IN THE BEGINNING, BECAUSE WHEN IT BRUNS, IT MUST JUMP TO THIS
SMALL PRO G RAM FIRST.  NOW YOU CAN BOOT UP YOU 3.3 DISK, AND TYPE "CALL-151
(RETURN)", "9FD :  4C 00 34 (RETURN)","A964:FF (RETURN)", AND "BSAVE
GALAXIAN,A$9FD,L$8C03 (RETURN ) ", AND NOW YOU ARE FINISHED.

        AGAIN,BROUGHT TO U BY
       MR. MADNESS...........
     OF PIRATES OF THE ROUND TABLE
    "MAY PIRATING LIVE FOREVER!!!"

  :::: GENERAL INTEREST TOP
```

```
==============================================================================
DOCUMENT usr.16.8k
==============================================================================

              _________  __  ___    ___  ______________________
              \||||||||\ \/ /|||\__/|||\/||||||||||||||||||||||\
              /'''''''''\   \''''''''''/'''''''''''''''''''''''/
             /    /~\    \   \____    /     ~~~~~~/    ~~~    /
            /    /__/    /______/    /    _______/    _     _/
           /            /           /           /    / \    \
           \___________/\__________/\__________/\___/   \___/

     ______________  ___________  ___    ___  __  ______________________
     \|||||||||||||\/|||||||||||\/|||\  /|||\ \/ /||||||||||||||||||||||\
     /'''''''''''''/''''''''''''/''''/_/''''/   /'''''''''''''''''''''''/
    /    /   /    /    ~~~     /           /   /     ~~~~~~/    ~~~    /
   /    /\__/    /    ___     /    _    _/    /    _______/    _     _/
  /    /   /    /    /  /    /    / \    \   /           /    / \    \
  \___/    \___/\___/   \___/\___/   \___/   \__________/\___/   \___/

             _____  ________________________/\  _________________
            /  _  \/  ___/  ___/  ___/  ___/  \/  /__  __/   ___/
           /  ____/  /  /  ___/___ \/  ___/      /  /  / /___  /
          /__/   /__/  /_____/_____/_____/__/\  /  /__/ /_____/
                                              \/

                    ___________  _________
                   /__   __/  /_/  /  ___/
                     /  / /  __  /   ___/  __  __  __
                    /__/ /__/ /__/_____/  /_/ /_/ /_/

    __   __ ____________         ___
   |  | |  |     HHHHHHH         HHH
   |  | |  |  |  HHH HHH         HHH                   sss
   |  | |  |  |  HHH HHH  sssss  HHHsss   sssss   HHH  """  sssss   sssss
   |  | |  |  |__HHH_HHH HHH HHH HHH HHH HHH HHH HHHHH HHH HHH HHH HHH HHH
   |  | |  |\__ \HHHHHH" HHH HHH HHH HHH HHH HHH  HHH  HHH HHH HHH HHH HHH
   |  | |  |  |  HHH HHH HHH HHH HHH HHH HHH HHH  HHH  HHH HHH     "HHsss
   |  | |  |  |  HHH HHH HHH HHH HHH HHH HHH HHH  HHH  HHH HHH HHH sss HHH
   |  `-'  |  |  HHH HHH HHH HHH HHH HHH HHH HHH  HHH  HHH HHH HHH HHH HHH
   |_______|_____HHH HHH  HHHHH   HHHhH   HHHHH   HHH  HHH  HHHHH   HHHHH

             .----------------------------------------------.
             |                                              |
             |       INFO FILE ON 16,800 BAUD MODEL!        |
             |                                              |
             | Typed, Hacked & Investigated By: /X\R. YU/<! |
             |    Call D'YER MAK'ER BBS: 1-908-730-8633     |
             |                                              |
             |        A -*- NEMESiS -*- PRODUCTION!         |
             |                                              |
             `----------------------------------------------'
```

U.S. ROBOTICS SYSOP PRICING

PRODUCT	PRICE
COURIER HST (U.S. AND CANADA)	$ 399
COURIER V.32bis (U.S. AND CANADA)	$ 449

COURIER HST DUAL STANDARD (U.S. AND CANADA)	$ 499
INTERNATIONAL HST*	$ 389
INTERNATIONAL V.32bis*	$ 439
INTERNATIONAL COURIER HST DUAL STANDARD*	$ 449
DOMESTIC POWER SUPPLY (INT'L USERS)*	$ 10
INTERNATIONAL POWER SUPPLY (INT'L USERS)*	$ 50

*International version and a separate power supply must be ordered if the modem is used outside the U.S. or Canada. International power supply is 220 volts. Domestic power supply is 110 volts. At least one (1) power supply must be ordered with each international modem.

For detailed information and technical specifications on the above products, call the U.S. Robotics Technical Support Department at (800) 982-5151 in the U.S. or (800) 553-3560 in Canada. From outside the the U.S. or Canada, call (708) 982-5151.

```
.-------------------------------------------.
|                                           |
|    T A B L E   O F   C O N T E N T S !    |
|___________________________________________|
|                                           |
|  PART 1  -  SPECIAL INTRO    BY: MR. YUK  |
|  PART 2  -  EDITORIAL        BY: MR. YUK  |
|  PART 3  -  MESSAGE CAPTURE  BY: MR. YUK  |
|                                           |
`-------------------------------------------'
```

```
-----------------------------------------------------------------------------
                      ___  ___  ___  ___     ___
                     |   ||   ||   |` | '     |
                     |---'|---||-+-'  |       |
                     |    |   ||  \   |      _|_
```

INTRODUCTION!

Well around Febuary of this year there was much talk about a US Robotics COURIER HST, with a new speed mode. There were some lame texts coming out that explained shit! So I went on a quest, I called USR, talked to techies, talked to some dudez that heard these rumors, and logged on various boards gathering information. So here it, all in a very big text file. But at least I'm sure it'll answer all your questions on the new HST's. Also another note is that these NEW HST's are the same price.

Look for more texts like these on my board, D'YER MAK'ER. Please don't edit, add or change anything in this doc file. ENJOY!

```
-----------------------------------------------------------------------------
                      ___  ___  ___  ___    _____
                     |   ||   ||   |` | '    | |
                     |---'|---||-+-'  |      | |
                     |    |   ||  \   |     _|_|_
```

Ok the NEW USRobotics COURIER HST with v.42 & v.42bis is available as of March 1992. The prices haven't changed, and special sysop deals can let you have the fastest modem in the world for only $399!!

That's right! The new HST has (this is according to rumor) a 88c188 processor running at 33mhz. Speeds supported are: 300,1200,2400,4800,7800, 9600,12200,14400, & 16800. The new modem can be locked at 57,600baud, this is also a new feature. The modem is 2 inches smaller, and has a nicer design. SYNCHRONOUS transmission has been stepped up from 450bps on the back channel to 1200bps. Many new commands are installed, also a v.54 link diagnostics, many testing routines, and special commands for UNIX users. More ATI? commands. Special expansion for the future is built in (Like the when IBM came out, and was built for expanding).

Well in my opinion this is the most impressive piece of computer hardware ever built. This modem is worth it's weight in gold, and I've already ordered one. I also have a sneaking suspicion that anyone who is anyone will have one of these. It's like when the 14.4kbps replaced the 9,600kpbs. This will replace the 1440, and boards (that are always busy) will soon only support 19200 to 57,600 baud only, and 9600 will be dead. Well, all can say, is try to order one of these as soon as possible, because they are selling fast (there is already a waiting list, according to a USR Representative).

Now read the messages from the tech. section on the USRobotics tech support section, that I captured for ya! ENJOY!

```
------------------------------------------------------------------------------
                        ___  ___  ___  ___   _______
                       |   ||   ||   |` | '   | | |
                       |---'|---||-+-'  |     | | |
                       |    |   ||  \   |    _|_|_|_
```

THESE ARE ALL THE MESSAGES I RAPED OFF OF THE TECH SUPPORT BOARD! -/X\R. YU/<
I EDITED THE TEXT, AND PUT LINES IN BETWEEN THE MESSAGES FOR EASIER READING!

```
Msg#: 1625 *USR Tech Support*
03-03-92 22:30:50
From: BRIAN HOAG
  To: NICK DODGE (Rcvd)
Subj: USR 16.8K
```

Nick,

Can you please answer me one question? Is the current Dual Standard that is shipping the one that runs 16.8k? If so, I want one, but if not, I will wait. Please tell me what the deal with these is. Thanx!

Brian

```
Msg#: 1627 *USR Tech Support*
03-04-92 04:50:21
From: MATS JEBORN
  To: SYSOP (Rcvd)
Subj: 16.8
```

Hi, ive heard rumors that the new 16.8 modem i out?! Is that a fact? How much will the price be? Will the other modems be cheeper or will they go out of stock? Sincerely, Mats

----

Msg#: 1696 *USR Tech Support*
03-04-92 14:33:42
From: NICK DODGE
  To: BRIAN HOAG (Rcvd)
Subj: REPLY TO MSG# 1625 (USR 16.8K)

Brian, all of the Sysop modems currently shipping are the 16.8 model. Hope this helps.

Msg#: 1698 *USR Tech Support*
03-04-92 14:37:28
From: NICK DODGE
  To: MATS JEBORN (Rcvd)
Subj: REPLY TO MSG# 1627 (16.8)

Mats, yes, all of the modems shipping out of the USR Sysop program are 16.8. The prices have stayed the same. Hope this helps.

Msg#: 1704 *USR Tech Support*
03-04-92 20:46:30
From: DAVE PINNER
  To: SYSOP (Rcvd)
Subj: 16.8K MODEMS

I'm using a 14.4EX Intel modem right now, but I'm quite interested in your 16.8 model. Is this essentially a v.32bis modem with v.42bis and some enhancements you've made to whip it up to 16.8? I'm not familiar with an industry spec for a 16.8.

Another nosey question.... Your logon bulletin indicates that you have the ports for this BBS locked at 19,200. Is that a limitation of some of the hardware you're using in the configuration? I would've figured you'd lock the ports at (what I was told was the maximum port speed for the USR Dual), 38,400.

Nice BBS... Classy! I'm impressed!

Msg#: 1708 *USR Tech Support*
03-05-92 00:40:01
From: BRIAN HOAG
  To: NICK DODGE (Rcvd)
Subj: REPLY TO MSG# 1696 (USR 16.8K)

Yes, it does help, Nick. Now I know I can buy one! BTW, I forgot ta ask ya, if the old dual will do 19.2k, what will the 16.8k one do? 24k? How would that setting work with a communications package (GT Power) that only has settings at 14.4 and 19.2? Thanx much for your help! Brian

----

Msg#: 1906 *USR Tech Support*

03-05-92 18:21:26
From: JIM HOJNICKI
  To: TECH SUPPORT
Subj: 16.8K MODEMS

I Have an earlier Courier V.32bis modem and was wondering if there will be an upgrade to this new higher speed standard (16.8kbs). Could you provide a few details about this new standard (besides the fact that it's faster).

Also, last week I left a message about lockups I was experiencing during downloads (ZMODEM and YMODEM-G) with my Courier V.32bis modem. I have yet to receive a reply! In the meantime I upgraded my UART chip to a NS16550AFN and am still experiencing lockups. The same software on an IBM PS/2 and my Dual Standard (V.32bis/HST) does not lock up, yet even with the 16550 chip I still lockup at home. Are there known bugs in the very early models of the Courier V.32bis modems that can be corrected by a ROM (or other) upgrade?????????????

Msg#: 2026 *USR Tech Support*
03-10-92 17:54:28
From: ED TAGGART
  To: TECH SUPPORT
Subj: HST DS > 16.8

I recently purchased a USR HST DS (less than 1 month ago), through the syso program. I now understand that you are shipping a new HST DS capable of communicating at 16.8k bps. Will there be any upgrades available to recent purchasers of your modems? Or some sort of ROM upgrade? I would have waited a few weeks or so if I had known that there was an enhanced HST DS coming out. I patiently await your reply...

Msg#: 2085 *USR Tech Support*
03-11-92 19:55:57
From: ED TAGGART
  To: TECH SUPPORT
Subj: HST DS > 16.8K

I was wondering if I could get an answer to the message that I posted yesterday reguarding upgrades for for the HST DS to the new 16.8k bps modem? You can call me at 207-799-1138 Thanks!

Msg#: 2109 *USR Tech Support*
03-12-92 13:11:23
From: LIANG-KUAN YEH
  To: ED TAGGART
Subj: REPLY TO MSG# 2085 (HST DS > 16.8K)

        Yes...I would like to know that exact thing....Has USR reply to you before? I wrote a message about 1 week ago and no reply regarding the new 16.8k modems....if you find any info on a upgrade or anything...could you please drop my a line...thanx..

```
Msg#: 2114 *USR Tech Support*
03-12-92 15:41:47
From: JOHN LESTER
  To: ED TAGGART
Subj: REPLY TO MSG# 2085 (HST DS > 16.8K)

Heh...since I know this information, I figure I'll post..
. The NEW 16.8 modems are a TOTALLY different 'breed'..they are smaller
physically....and there is NOT upgrade option to get 16.8 if you have an 'old'
dual or HST...
. Kinda sucks if you bought your modem a month ago...I don't know if USR is
taking 'returns'....I doubt it, tho...
. Now..the BIG question is...when v.fast (19.2K) becomes a reality in a couple
years or so...will the new 16.8 modems (duals, HSTs) be upgradable?....heh heh
heh....

------------------------------------------------------------------------------

Msg#:  779 *US ROBOTICS*
02-13-92 02:13:00
From: DAVID BERNARD
  To: ALL
Subj: HST UP-DATE TO 16.8

This is from RIME Conference Host Jim Daly announcement:

I made mention a month or so ago of some new products forthcoming from
the folks in Skokie.  I can now give you the details.

There has been some concern amongst the end-Users who had invested in
the HST-14.4 protocol that USR had abandoned them in favor of the new
v.32bis standard.  Not true!

US Robotics strikes again with a new HST-14.4 protocol!!  Beginning
immediately (product is already in the pipeline), the new HST-14.4 wil
now support transfers at 16,800.                      ^^^

The new Dual Standards will also support 16,800 on the HST side.
    ^^^
 It is important that everybody understands that the 16,800 can only be
 accomplished between the NEW HST-14.4 and DS Modems.  There is NO
 UPgrade Path nor is one planned!  ie: if you have a earlier version
 (meaning any unit shipped prior to 2/1/92) of the 14.4 or DS the
 maxim connect to the NEW models will be 14,400.

All prices at the Retail Level and the various Demo programs will remain
the same....no increases anticipated.

P.S.  The entire Courier line has been redesigned and "DOWNsized".

David says!
  NOTE: My comment to it is, I as a HST owner/user am again left on
  the outside with no up-grade path possible.  It may lengthen the
  life of HST and add to USR bottom line but how about current owners!
  I feel sold out with no support from USR!  Again sell the old and
  buy a new one from USR!  Hmmm!  FYI Later...David
```

```
------------------------------------------------------------------------------

Msg#: 1016 *US ROBOTICS*
02-14-92 15:58:00
From: BILL UTTER
  To: DAVID BERNARD
Subj: REPLY TO MSG# 779 (HST UP-DATE TO 16.8)

  >  NOTE: My comment to it is, I as a HST owner/user am again left on
  >  the outside with no up-grade path possible.  It may lengthen the
  >  life of HST and add to USR bottom line but how about current owners!
  >  I feel sold out with no support from USR!  Again sell the old and
  >  buy a new one from USR!  Hmmm!  FYI Later...David

        But when you bought a 2400 was there a path to upgrade it to a
 9600.  If you have a 9600 V.32 is there a path to upgrade it to a 14400
 V.32bis.  So that's just life.  When you bought your 286 was their a
 path to upgrade it to a 386, etc, etc, etc.

------------------------------------------------------------------------------

Msg#: 1030 *US ROBOTICS*
02-15-92 08:16:00
From: HARDY ROSENKE
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1016 (HST UP-DATE TO 16.8)

* At 02:13 on 92-02-13, David Bernard wrote to All ...
DB> This is from RIME Conference Host Jim Daly announcement:
  ....  DB> now support transfers at 16,800.                    ^^^
DB> The new Dual Standards will also support 16,800 on the HST
DB> side.
       Thanks for the info!  It is both TIMELY and well received <grin> as my
order is in the pipeline and my cheque was cashed yesterday!! ... good to know
I am getting a late "birthday" gift from USR!

------------------------------------------------------------------------------

Msg#: 1053 *US ROBOTICS*
02-17-92 02:17:00
From: DAVID BERNARD
  To: BILL UTTER
Subj: REPLY TO MSG# 1030 (HST UP-DATE TO 16.8)

|        But when you bought a 2400 was there a path to upgrade it to a   |
| 9600.  If you have a 9600 V.32 is there a path to upgrade it to a 14400 |
| V.32bis.  So that's just life.  When you bought your 286 was their a    |
+-------------------------------------------------------------------------+
 Hey I agree that is life, but there are modem companies (Intel & Hayes)
 that I know of that in deed did offer a reasonable up-grades to their
 purchasers/users from V.32 to V.32bis!  IMHO no matter what US R says
 it seems to be a corporate decision for their bottom line not to offer
 it so they can sell more modems.  Hey when they were the only game in
 town they could get away with it, now there are other cheaper options!

 I personally hope HST hangs around a lot longer but I got my warning and
 learned my lesson new high tech from US R means no up-grade it has
 happened so many times.  So the new HST 16.8 comes out and 6 to 12
```

 months from now they get a faster better standard, history tell me US R
 again will have no up-grade path, it is simple, Buyer Beware & know
 what you are going to face in the long run. Knowing that then make
 your decision as you may. But just look at US R's past history and you
 will see what I mean! Hey, I am a fan but this is not the way to
 treat fans by allways leaving us out in the cold! Ltr...David

----

Msg#: 1074 *US ROBOTICS*
02-17-92 07:51:00
From: TOM HENDRICKS
  To: HARDY ROSENKE
Subj: REPLY TO MSG# 1053 (RE: HST UP-DATE TO 16.8)

```
 > * At 02:13 on 92-02-13, David Bernard wrote to All ...
 > DB> This is from RIME Conference Host Jim Daly announcement:
 >   .... DB> now support transfers at 16,800.
 >                   ^^^
 > DB> The new Dual Standards will also support 16,800 on the HST
 > DB> side.
 >         Thanks for the info!  It is both TIMELY
 > and well received <grin> as my
 > order is in the pipeline and my cheque was cashed
 > yesterday!! ... good to know I am getting a late
 > "birthday" gift from USR!
```

As far as I know they haven't been officially released or announced by USR yet,
hold on to you hat and see which model actually arrives.

-Tom-

Msg#: 1094 *US ROBOTICS*
02-15-92 21:38:00
From: JIM BEDICS
  To: ALL
Subj: NEW 16.8K (OR WHATEVER) QUESTION......

     I just got my 14.4k DS last week, and found it was DOA. So, I sent it
back overnight airmail insured which cost me $40 (which I wasn't too happy
about), to have it serviced. Now, here is my question. If there was a problem
in my chips (where I think the problem was) do you think I will get the new
16.8k chips in there? Or, will they just give me some used chips. Question
#2. Has anyone else ever sent their modem back to USR for a SERIOUS problem (I
m sure somebody has)? Did they "fix" your old modem and send it back, or give
up, and send you a new one. I think mine was beyond hope, as all it did was
blink at me (MR and CD LEDS) 100% of the time. I was hoping they would send me
a new one (which would HOPEFULLY be the 16.8k) but wasn't sure what their
policy on returns/serviced modems was. Thanx for any info.

Msg#: 1127 *US ROBOTICS*
02-18-92 12:59:00
From: STEPHEN HENDRICKS
  To: BILL UTTER

Subj: REPLY TO MSG# 1074 (HST UP-DATE TO 16.8)

 On 02-14-92 Bill Utter wrote to David Bernard...

```
 BU>   >  NOTE: My comment to it is, I as a HST owner/user am again left on
 BU>   >  the outside with no up-grade path possible.  It may lengthen the
 BU>   >  life of HST and add to USR bottom line but how about current
 BU> owners!
 BU>   >  I feel sold out with no support from USR!  Again sell the old and
 BU>   >  buy a new one from USR!  Hmmm!  FYI Later...David
 BU>
 BU>         But when you bought a 2400 was there a path to upgrade it to a
 BU>  9600.  If you have a 9600 V.32 is there a path to upgrade it to a 14400
 BU>  V.32bis.  So that's just life.  When you bought your 286 was their a
 BU>  path to upgrade it to a 386, etc, etc, etc.
```

Are we to be slaves to old technology? Progress does not come with a guarentee that obsolete technology is going to be adaptable. It is an absolutely ludicrous idea that USR should have any responsiblity to upgrade OLD technology when they spend $Millions every year to develope new techniques.

When was the last time GM offered to put a new engine in your car because the new one had more horsepower and gets better fuel economy? You buy a new car to get new technology.

The aforementioned buyer should view his modems obsolescence in the manner which is most benefiscial to him. Namely that because he owns a USR HST modem it's resale value is great enough to allow him to upgrade (buy selling it and buying the new modem) at the cheapest possible price. This method of upgrade ensure that no risk in incurred with changing of processors, and no delay (other than the ordering delay) occurs if the modem were returned to the factory for rebuild. It is precisely because the USR modems have the greatest acceptance in the marketplace for high speed modems that the investment is secure.

Let's get realistic in this view. The price that an older HST modem can be sold for, to people still using 1200 and 2400 modems (or practically any other High Speed modem for that matter),
allows the purchase of new technology modems in a safer and more orderly fashion than trying to keep track of the dozens of old versions that have have been produced for the purpose of upgrading them. Unless an upgrade is a firmware only proposition, the cost is astronomical.

The fact that the USR modem has such wide acceptance is your best guarentee that your money is wisely invested in ANY USR HST modem.

Use the HST-Sale echo for this purpose.

Msg#: 1133 *US ROBOTICS*
02-18-92 14:00:00
From: STEPHEN HENDRICKS
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1127 (HST UP-DATE TO 16.8)

 On 02-17-92 David Bernard wrote to Bill Utter...

DB> Hey I agree that is life, but there are modem companies (Intel & Hayes)
DB> that I know of that in deed did offer a reasonable up-grades to their
DB> purchasers/users from V.32 to V.32bis! IMHO no matter what US R says

Like Groucho you came up with the magic word that both explains your lack of understanding and why it is NOT reasonable to upgrade many HST modems. During the last few years, and while maintaining a similar outward appearance the HST modems were completely redesigned from analog to digital devices. In some cases upgrades were possible. However, unlike Hayes, USR has over 100000 of these modems in the field and several different types. Unlike Intel and Hayes, the USR modems have a VERY high resale value. It is more practical in most instances, even when there was an upgrade available (except maybe adding V.42bis to V.42), to sell and rebuy to attain the NEWEST technology. If you have a Hayes 9600V, what upgrade path do you have to V.32? Have you seen the Hayes 9600 V.32 that costs over $1500, did Hayes upgrade that modem? I think not. Hayes advertises it's 9600 Ultra as compatible with every high speed modem in the world, but it fails as it won't talk HST, is Hayes going to upgrade it?

I agree with your sentiment, but find that no other company equals USR support.

DB> it seems to be a corporate decision for their bottom line not to offer
DB> it so they can sell more modems. Hey when they were the only game in
DB> town they could get away with it, now there are other cheaper options!

Actually you do very well on this, cheaper yes! But value: NO! Remember the 9600V, remember the Hayes 9600 V.32, remember Microcom's MNP10. Better yet remember the Compucom!

If as you suggest USR was to maintain an upgrade path for older technology modems, are you willing to take the blame for no forthcoming higher performance levels. Having to hobble technology in
the guise of upgradability would condemn any company to mediocrity. Have you seen DrDos 6.0 as compared to MS/Dos 5.0? Maybe you don't see the similarity in this comparison, but it is likely that mediocrity would win if your view was prevalent.

Msg#: 1134 *US ROBOTICS*
02-18-92 14:09:00
From: STEPHEN HENDRICKS
To: DAVID BERNARD
Subj: REPLY TO MSG# 1133 (HST UP-DATE TO 16.8)

DB> Hey Hardy, Apparently, in your case you will do OK. But you will also
DB> have to wait for others to get the new one to use the little extra
DB> 2400/bps, again if US R would offer up-grades to the rest of us we also
DB> could be on even keel at that speed and it would add a better feeling

You don't mention a damned thing about the higher speed being given to new customers AT NO HIGHER Price. Equally you fail to mention that because of USR innovation the High Speed modem came into being. And you fail to note that USR has hundreds of thousands of these modems in the hands of average users, not corporate mis departments. Keep on kidding yourself that you have been badly used by USR in not catering to the obsolete technologies of last year and you will sooner or later convince yourself that some other modem would do as well. I tried and nearly fooled myself too!

You don't know how much trouble there has been with other modems that use
what I consider to be deliberately misleading advertising. One competitor
ALWAYS uses LINK speed instead of carrier speed in his ads. Another
claims TOTAL modem compatibility, but won't even talk to his own brand
9600 modem or the HST, the world best accepted High Speed modem.

Instead of being satisfied that you have a modem that has genuine resale
value, and is compatible with 100% of the existing modems you moan about
not being compatible with a VAPORWARE modem.

What a crock!... or maybe an alligator!

```
Msg#: 1153 *US ROBOTICS*
02-19-92 02:19:00
From: DAVID BERNARD
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1134 (HST UP-DATE TO 16.8)

|You don't mention a damned thing about the higher speed being given to|
|customers AT NO HIGHER Price.  Equally you fail to mention that becaus|
|USR innovation the High Speed modem came into being.  And you fail to |
|that USR has hundreds of thousands of these modems in the hands of ave|
|users, not corporate mis departments.  Keep on kidding yourself that y|
|have been badly used by USR in not catering to the obsolete technologi|
|of last year and you will sooner or later convince yourself that some |
|other modem would do as well.  I tried and nearly fooled myself too!  |
+----------------------------------------------------------------------+
 I will start off with a Love my US R Dual HST/V.32!  But!!!  16.8 HST?
  OK the highest non-standard Tech with a street price at 575+ to 775+,
 Gee folks will be breaking down the doors to get them as opposed to the
 newer V.32bis modems at or below 500?  The thousands of HSTs out there
 can't be up-graded like I said, my old dual can't be & I am glad that
 US R will try to keep HST around for a while but even getting the new
 16.8 you have to wait till everybody else gets it to do any faster.
 So what is the advantage to 16.8, it has no future since it is not
 standard & only limited current use when others get it, and at that
 cost too!  How many 16.8 HST only modems will be in demand?  So based
 on past history of US R it may be assumed that these also won't be able
 to be up-graded to the next higher speed?  That makes it a Lame Duck!

 Now if they come out and said we will make these new modems available,
 and we will make the Dual up-gradeable to the next faster standard at
 reasonable cost, and sell it at a competitve street price then I would
 venture to say US R could blow all those others guys out!  FYI:

Again from RIME:
|I guess the word "downsized" threw a few folks.  Their terminology, not    |
|mine.  The production code word was v.SMALL                                |
|                                                                           |
|What is meant is that the new design is smaller.  I've not seen one yet    |
|so I don't know how much smaller.  COurier will remain the premium         |
|category in their lineup of product:                                       |
+---------------------------------------------------------------------------+
```

I don't feel I have been used by US R, I just will make my decision
on the next modem based on the Tech available, price, and the ability
to up-grade in the future for even faster if and when it comes out.
The lowest price is not the main element but a reasonable competitve
one is important. The message I get from US R is that they are more
interested in selling more new modems then up-grading present ones,
and that is fine but the Buyer Beware should be known and then do as
you/I may from that point. That is what I plan to be doing and any
consumer that really does their research and cares for the future I
believe will also doing a similar thing. I may in fact buy another
US R but I won't do it blindly just because it is US R, they will
have to earn my next purchase, and not live on their past reputation!
 Ltr..David

Msg#: 1374 *US ROBOTICS*
02-25-92 18:21:00
From: MIKE DRUMMOND
  To: LARRY NESBITT
Subj: NEW 16.8K (OR WHATEVER) Q

LNHH>MD=>JBHH>   I just got my 14.4k DS last week, and found it was DOA.

LNHH>   When I read your message..'Boy' did it hit the nail on the head!
LNHH>I had just purchased the HST 14.4 courier back in May of 89 and found that
LNHH>the modem would not run properly. So I followed the instruction and calle
LNHH>the support line. They were very curtious, pleasent, and understanding of
LNHH>my problem and gave me a shipment number. I retured this modem in the
LNHH>proper packaging and sent it first class 'thinking it will come back soone
LNHH>well time passed and I called the support toll-free 800 number and they
LNHH>refered me to the tech's that were working on the modem. Again they were
LNHH>very pleasent...but no modem. After three weeks, I finally got a notice
LNHH>that my system would be sent in a couple of days. I finially got it a
LNHH>week later 'second day service'. It cost me with shipping, handling,
LNHH>and insurance $38.50. Well after installing it...I thought things were
LNHH>great!!! Well, it didn't last but a month and the modem did the same thin
LNHH>so I call the support people up again...they checked it out and then gave
LNHH>a number........Well after sending it in a couple of other times they did
LNHH>correct the problem....it was a loose screew in the mother board of the
LNHH>modem. Well, to make this long story short...it finially cost me over $90
LNHH>and months of waiting and runing my system at the slow cps of 2400 baud.
LNHH>   I really do like the HST and will by another, but your story struc
LNHH>a bell and I thought I would mention mine. Better luck in the future!

LNHH>   Larry....

Well you have more patience than i do. I bought a ZyXel modem and the
support they provide is outstanding. I am afraid that i have bought my
last USRobotics modem. They sat on there behinds to long patting
themselves on the back while the compitition left them behind
(Concerning service anyways).

Msg#: 1376 *US ROBOTICS*
02-23-92 10:52:00
From: MANUEL WENGER

  To: ALL
Subj: HST 16.8K ETC

Hi everybody!
I've read some messages in this area about the new HST (and Dual) which is smaller and has the HST 16.8K modulation. I've called the USRobotics BBS, but even their BBS only supports HST 14.4K (and of course V32bis etc). Now, is this HST 16.8K only a "rumor" or is it true? And HOW MUCH does it cost if I DO NOT buy it with the Sysop Deal? And WITH the sysop deal? And is an upgrade from an...er.."old" Dual to the new one with HST 16.8K possible or not?

byby
  Manuel

Msg#: 1383 *US ROBOTICS*
02-25-92 13:34:00
From: KLAAS HAMBOERGER
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1153 (HST UP-DATE TO 16.8)

Hello,

```
 > |RE: the above subject line..I called USR yesterday and they  |
 > |said there was no fastermodem available at this time. The    |
 > |only changes at present was SIZE ONLY of both the DS and     |
 > |the HST.                                                     |
 > +-------------------------------------------------------------+
```

As far as I know, the v.small models have only as much horsepower as the current models: A 80188/16 and the one ore two signal processors. Is that right? If so, the "older" duals mit v.32bis should be able to deliver the 16.8k HST-speed. If this is true, it must be possible for USR to offer an update by only changing the ROMs. If this is right, USR only doesn't want to update, although they could. If not, the new models must have some improvement in processing speed. Does anybody have the exact specifications? If the new models are as slow as the current ones and USR doesn't offer an update, they just are to "lazy" or so to make the firmware of the new generation fitting for the modems you get today.

Something to the car-example: If a new car is being developed and you are interested in cars you know at least half a year before the official announcement that it will come, because you can read about it in car magazines. Another fact is, that you know how long the current model is on the market. If it was released in 1990, you know for sure, that there will be no successor before 1994 or later. So you can plan exactly when it is wise to buy a new car. With modems this is completely different. My dual standard was delivered on the 10th of january. When I ordered, I thought, that my new modems would be up to date for some time. At this time USR said, that no new model would be released in the next time. If I had known, that they were already going to release the v.small-models in february/march I wouldn't have ordered at this time! I think, that USRs behaviour isn't very fair. It was hard to save enough money to buy the modem. Now it is only one month old and already antiquated.

               Ciao, Klaas

Msg#: 1428 *US ROBOTICS*
02-27-92 11:46:00
From: STEPHEN HENDRICKS
  To: MANUEL WENGER (Rcvd)
Subj: REPLY TO MSG# 1376 (HST 16.8K ETC)

 On 02-23-92 Manuel Wenger wrote to All...

 MW> I've read some messages in this area about the new HST (and
 MW> Dual) which is smaller and has the HST 16.8K modulation.
 MW> I've called the USRobotics BBS, but even their BBS only
 MW> supports HST 14.4K (and of course V32bis etc). Now, is this
 MW> HST 16.8K only a "rumor" or is it true? And HOW MUCH does

We play a game every year with US Robotics where they announce at Comdex or some other venue that they intend to come out with a new product. Then a variety of "reliable sources" start to give out contradictory information about dates, specifications etc. Last week USR has dropped the word that the new modems are supposed to ship on 3/2/92. But they also said they would ship on 2/1/92. So this is an announcement that could prove as false as the previous one. I suspect they will delay until distributor stocks are down, and the new modems are actually needed for shipments to distributors.

 MW> it cost if I DO NOT buy it with the Sysop Deal? And WITH
 MW> the sysop deal? And is an upgrade from an...er.."old" Dual
 MW> to the new one with HST 16.8K possible or not?

Sysop's deal on a Dual Standard is $499. This is not an upgrade of an older modem. It is a completely NEW design, with 16800 dce and 57,600 dte rates. It is also approximately Half the size of the older modems.
There have been more than a dozen different USR HST modems, most bear the same name as the predecessor and are in reality radically different inside. When massive changes are made, there is no realistic expectation of an upgrade.

Currently USR HST modems have a VERY High resale value. Judging from the HST-Sale echo, the modems are in great demand. If you have an older modem, there will be no difficulty in selling it, and purchasing the new design for much less than the cost of a "so-called" upgrade.

Msg#: 1430 *US ROBOTICS*
02-27-92 12:00:00
From: STEPHEN HENDRICKS
  To: MIKE DRUMMOND
Subj: NEW 16.8K (OR WHATEVER)

 MD> Well you have more patience than i do. I bought a ZyXel modem and the
 MD> support they provide is outstanding. I am afraid that i have bought my
 MD> last USRobotics modem. They sat on there behinds to long patting

Then you have no reason to be entering messages in this echo. Start you own if you like, but you have no need of the technical support this echo was founded to provide.

Msg#: 1434 *US ROBOTICS*
02-27-92 12:15:00
From: STEPHEN HENDRICKS
  To: KLAAS HAMBOERGER
Subj: REPLY TO MSG# 1383 (HST UP-DATE TO 16.8)

 KH> would be released in the next time. If I had known, that
 KH> they were already going to release the v.small-models in
 KH> february/march I wouldn't have ordered at this time! I
 KH> think, that USRs behaviour isn't very fair. It was hard to
 KH> save enough money to buy the modem. Now it is only one
 KH> month old and already antiquated.

The new modems were talked about at Comdex last fall (November ?) So for
those with access to industry wide news, it was no surprise. USR is like
Porsche, they are constantly making improvements without regard to model
years. If you have been involved with automobiles, especially in the USA
the makers constantly change the internal components. A typcial auto made
in one year might have four different make axles in it for example.

It is probably a greater problem to upgrade in Europe than here, but we
have little problem upgrading from one modem to the next due to a very
high resale value. Sorry that this happens, but I think it is best that
as soon as new technology is available, it be introduced.

The V.Small series is half the size, with different construction than some
previous modems. If the new DCE rate is 16,800.. there is no guarentee or
even a reasonable expectation that a rom upgrade would provide a similar
improvement in an older design. USR has made the V.42bis upgrade
available at very reasonable cost in a past upgrade which involved
software only. This is not an upgrade to V.Small but an entirely new
modem series, and as such it is not reasonable to expect USR to redesign
an older series to equal a new design.

Do you think Porsche would upgrade existing 356s?

Msg#: 1456 *US ROBOTICS*
02-26-92 14:15:00
From: HARDY ROSENKE
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1434 (HST UP-DATE TO 16.8)

DB> Hey Hardy, The past is the past, but the future is what I will
DB> spend my hard own money on for a my next modem. If US R holds to
DB> the no up-grades for the next line to come that is fine but I won't
DB> spend it on 16.8 HST with the known fact that faster is coming.
DB> Since no current HST can't be up-graded to 16.8 and if the new
DB> line won't up-gradeable (maybe it will!) then IHMO what USR is
DB> doing is trying to hold on to market position with a stop gap
DB> measure, and I choose not to buy it. If by chance they do come out
DB> with 16.8 and say that the new one will have an up-grade path to the
DB> next higher speed at a reasonable cost then I would get one ASAP!
       Okay, I think that we are in agreement here. Perhaps what is needed is
for current owners of USR products and all 'potential' owners of USR products

to put a little pressure on USR to make sure that the upgrade path is available. I would certainly think that they would want to stay competitive, offer SysOps good deals and keep the consumers happy! I share your sentiments.. what good is a 16.8K modem if it has no one to connect with at that speed?? I have used that arguement on people that have asked my advice on buying >choke< CompuCom modems -- what the heck would they connect with??? Perhaps it is time for owners of USR products to actively lobby USR to make and continue to make models that are upgradeable so that "Joe Average" does not have to sell his USR DS 16.8K modem in 1994 and go out an buy a new 100K USR Triple Standard.....

DB> I am glad US R is trying to keep HST around, but I don't think the
DB> masses will buy it at the latest prices.

This is true, but looking out there at what the users/sysops OWN, it is fairly obvious that HST is here to stay for a while, or people's HST's will start connecting at only 2400 if people start switching to cheaper brands. Granted, I would like to see price reductions to see more of my users being able to connect at high speeds to me, and I would like to call more other boards at high speed, but where will that speed come from? I want cheaper prices, but I do not want to have to buy 3 or 4 DIFFERENT modems to be able to take advantage of all of the protocols out there.... it is as insane as "standardization" [lack of] of BBS and mailer software.... or archivers for that matter.... everytime a new one pops up, WHAM! There goes at least another 200K of drive space.....

DB> not meet consumer demand. I hope US R does & what they will be
DB> offering soon will be attractive to the modem public in both Tech &
DB> Price, IMHO there past marketing plan needs adjustments for the current
DB> & soon to be competition. Hey, they can continue as is and they will
DB> sell modems but they will lose market share & that equates to lost
DB> bottom line.

This is very true.... perhaps they should be talking to you as a potential "SALES" person! <grin!> I would love to see their prices drop say 10%.... that would, even though it is not much, be a start ..... I do feel, however, that quality is worth paying a little bit more for. I can buy an "ABC-brand" modem for half the cost, but if it lasts only a third of the time as a USR, and is down all the time, then I would rather have spent more upfront for a better product!

Hardy

Msg#: 1457 *US ROBOTICS*
02-26-92 14:32:00
From: HARDY ROSENKE
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1456 (HST UP-DATE TO 16.8)

SH> You will be allowed to buy a modem/s under specified conditions for
SH> less than many dealers can buy a modem for. You will have NO dealer
SH> support whatsover, other than what a dealer feels like doing for free
SH> (usually nothing).

I was well aware of this, but I feel that you are unnecessarily painting all dealers with the same brush. I have talked to several out here that have gone out of their way to help SysOps with 'tempramental' USR modems.... and that seems to be more of the norm than the exception. Granted, they will not "replace" a modem for you, but they WILL send it out for repairs

thru their channels [which often times beats the send in service]
and will charge less money than POSTAGE for it... Maybe this is not so in the
US, but one thing is for sure... sending stuff thru the mail in Canada and
insuring it costs an arm and two legs....

SH> Unlike the normal customer your modem will be delivered out of USR
SH> Stocks in Chicago. This means that you will receive the most up to
SH> date modem that is being manufactured. You will not get a one year old
SH> model out of some warehouse, as you could buying as the public does.
SH> You will not necessarily get prompt service. You are not dealing with
SH> a an organization that has a sale staff that caters to individuals.
 That is an interesting fact which I did not know... I thought that they
would be pushing out their old inventory before the new, regardless of WHO or
WHAT the customer was. Granted that USR is not set up to deal with
individuals (per se) but they DO have a department to deal with SysOps, which
in my opinion is nice. The service seems to be prompt enough though,
especially when paying my personal cheque....it is nothing longer than would
be expected ordering thru a mail-order house....

SH> Too many people fail to appreciate the several hundreds of dollars
SH> they save, and the fact that they are circumventing the normal support
SH> procedures! If you want better turnaround time on orders, then buy
SH> from your dealer, like NORMAL people.
 First: *_I_AM_NOT_NORMAL!!_* Second: I do not like the tone that is
inferred in this, and I hope that you did not mean it the way that I am
reading it.... Third: "circumventing normal support procedures"??? Well, let
me just say that they had BETTER support my modem just the same as anybody
elses!!! Hundreds of dollars? Yes, I am saving that, but mainly becuase I am
buying direct.... not paying shipping costs that a DEALER would incur... I am
also not paying a stocking fee or a warehousing fee and the money I am paying
is not going into a salesman's pocket! I worked in the computer industry and
a markup of 40-50% on certain products is the NORM!! I can safely say that I
am paying only slightly more for my modem direct from USR than a dealer
ordering a GROSS of them would pay.... Turnaround time, yes, I could go out
and buy a DS this afternoon for $800... or wait here and get one shipped to me
from USR for $550..... I can wait.

Hardy

Msg#: 1459 *US ROBOTICS*
02-28-92 02:28:00
From: DAVID BERNARD
 To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1428 (HST 16.8K ETC)

|Currently USR HST modems have a VERY High resale value. Judging from
|HST-Sale echo, the modems are in great demand. If you have an older
+--
 Hey Stephen, Where is this confer available? Maybe some special deals
 to come if 16.8 comes out & SYSOPs want to up-grade, but we don't get
 that confer around New Orleans & I have heard others asking too.
 Well thanks for the help. Ltr....David

Msg#: 1465 *US ROBOTICS*

02-26-92 23:14:00
From: TOM SMITH @ 930/201
  To: MANUEL WENGER (Rcvd)
Subj: HST 16.8K ETC.

Manuel, the new "V.Small" USRobotics Courier line's more than a rumor. There's some indication that it's shipping now. Others indicate that it may not actually make it into the supply pipeline until late March. I guess we won't know for sure that they're out there until someone posts that his has arrived.

As for the prices, they're currently the same as the "current" Courier line. This'd mean that the HST Dual Standard, which's the only modem with HST that I'd recommend buying new, would run you about $500 through the SysOp Deal and about $750 retail. If you buy the latter, make sure that the dealer you're dealing with has the new ones in stock and won't be shipping you one of the current models. Sorry, but USR's not offering any kind of an upgrade to current HST users. I'd suggest that you hold off on buying one until next year's V.fast models come out. I don't think that you'd find the extra 2,400 bps in HST mode worth what it'd cost you to upgrade. In fact, until lots more of them ship, it'd not do you ANY good at all...

Tom Smith/Dallas...

Msg#: 1476 *US ROBOTICS*
02-27-92 13:25:00
From: TOM SMITH @ 930/1
  To: KLAAS HAMBOERGER
Subj: UP-DATE TO 16.8

Klass, the new "V.Small" USRobotics Courier line uses newer, more integrated, chips than the current line does. Its processors have significantly more "horsepower." Whether or not all of its power'll be fully "harnessed" at initial release only USR knows. There may be more tricks, such as V.fast, in that new box than we'll know about for some time to come. This increased processing capability also explains why a simple firmware upgrade won't take current Courier models up to the forthcoming 16.8 kb HST mode speed. Personally, I'm not a bit worried about that. Until a boatload of them ship, it's completely worthless in the first place. Second, the extra 2,400 bps doesn't impress me a bit. I'll be more than happy to truck along at 14,400 until V.fast ships in a year or so.

By the way, these new integrated circuits could explain why the clock speed's increased, too. We won't really know until some user gets his hands on one and reports on it in detail...

Tom Smith/Dallas...

Msg#: 1490 *US ROBOTICS*
02-28-92 09:30:00
From: BOB GERMER
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1457 (RE: HST UP-DATE TO 16.8)

 SH> The new modems were talked about at Comdex last fall (November ?) So for
 SH> those with access to industry wide news, it was no surprise. USR is like
 SH> Porsche, they are constantly making improvements without regard to model
 SH> years. If you have been involved with automobiles, especially in the USA
 SH> the makers constantly change the internal components. A
 SH> typcial auto made
 SH> in one year might have four different make axles in it for example.

In one YEAR? I have an 85 Horizon which we bought new. The left axle was made
by TRW. The right somewhere in Canada. I know because the constant velocity
joint boots differ depending on who made the axle. I had to have the left one
replaced because it tore and decided to do the right at the same time. The
mechanic ordered a left and right boot for the part number on the left axle.
The right boot wouldn't fit!

Bob

Msg#: 1491 *US ROBOTICS*
02-28-92 15:10:00
From: LARRY NESBITT
  To: MIKE DRUMMOND
Subj: REPLY TO MSG# 1374 (NEW 16.8K (OR WHATEVER) Q)

MD> Well you have more patience than i do. I bought a ZyXel
MD> modem and the
MD> support they provide is outstanding. I am afraid that i have
MD> bought my
MD> last USRobotics modem. They sat on there behinds to long
MD> patting
MD> themselves on the back while the compitition left them
MD> behind
MD> (Concerning service anyways).

      Like I said in the first message, I love my 14.4 hst modem and
wouldn't change for the world! Now, I must admit that I encountered
a problem with the service...but they stood by their product and sent
me at no cost for service and return mail the finished product. Where
I had a complaint was in shiping my modem to them....with insurance and
mailing not to mention packaging....it cost me (after four times) over
$90 dollars. This all could have been taken care of if the tech. would
have noticed the loose screw on one of the connections. I know it doesn't
sound very professional, but I can assure you that its still the very best
modem on the market and I for one will continue to use nothing but US
Robotics modems. Talk to you later and if your ZyXel modem is as good
as you say, how come I haven't heard of them? But that is in another
echo please...Or.....Net mail.

Larry......

Msg#: 1509 *US ROBOTICS*
02-28-92 02:28:00
From: DAVID BERNARD
  To: HARDY ROSENKE

Subj: REPLY TO MSG# 1490 (HST UP-DATE TO 16.8)

```
|HR>     Perhaps it is time for owners of USR products to     |
|actively lobby USR to make and continue to make models       |
|that are upgradeable so that "Joe Average" does not have     |
|to sell his USR DS 16.8K modem in 1994 and go out an buy a   |
|new 100K USR Triple Standard.....                            |
+-------------------------------------------------------------+
```

Well Hardy, I belive we both agree, but I also learned what USR's market thrust is and it is not the SYSOP/Users of BBS. I think that segmet is importnat to USR, but I don't think we have the $ to sway the company policy or their bottom line. As I see it USR can get their price from corp/bus/govt market and make good money from it, why should USR bend to the wishes of a much smaller market that does not really make a big contribution to their bottom line by offering to lower prices & offer up-grades to current modems when they can sell more new modems at their own prices to biz? Now I really don't know exactly what their sales marketing is, but I don't think they sell directly to dealers either in the chain and if that is so it adds to the eventual cost to the public too. Some of the other modem companies I have heard do sell direct to dealers & skip the distribution middle fellows.

I belive the BBSs segment is important to USR for the ad & name up-front on it & all the devoted people, it's much better then taking out ads in the PC Rags, that's my IMHO! I figure they don't lose on selling to SYSOPs direct since it does not go thru dealer net, but also most users can't afford to pay $550 to $800 for USR modems & IMHO USR really does not want those masses and will allow that segment to go to newer cheaper modems to come rather then lower their prices to theirs!

Did you see that business dude in here who complained USR would not sell him direct & who was willing to pay $1200 List to get 16.8 now, so we can't expect USR to listen to us about reducing prices. Hey the biz & govt customers just pass it along to the customers or taxpayers & they want faster speed now and not 1 or 2 years from now and will pay top $ for it now. I am not gonna fight or argue with it, I fully understand it but I don't agree with it but that is not gonna up-set USR if I don't buy one either. IMHO, I think a small price decrease similar to what you said is gonna come along as a bone for lower market and eventual cheaper V.32bis coming out, but they got the biz guys hot for the plucking now on 16.8 and they will make them pay for it!

BTW, we don't get voting rights with them unless you have some stock! I have been happy & still am with older Dual HST/V.32 and maybe I can get a deal on the "New old Dual" HST/V32bis from SYSOP who can't live without 16.8. I cried my eyes out already and now I realize where I fit on the list with USR's bottom line 1st. Good Luck.. Ltr...David

----

Msg#: 1577 *US ROBOTICS*
02-29-92 13:47:00
From: MIKE DRUMMOND
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1430 (NEW 16.8K (OR WHATEVER))

SHHH> MD> Well you have more patience than i do. I bought a ZyXel modem and the
SHHH> MD> support they provide is outstanding. I am afraid that i have bought m

SHHH> MD> last USRobotics modem. They sat on there behinds to long patting

SHHH>Then you have no reason to be entering messages in this echo. Start you
SHHH>own if you like, but you have no need of the technical support this echo
SHHH>was founded to provide.

I didn't mean to p@ss in anybodys wheaties. I was expressing my views on USRobotics. and while i do not plan to purchase from USR agian i do have a number of HSTs and therefore i do find this echo very helpfull. I like the product that USR puts out but the merchandise is only half of the formula the other half is support and this is where USR falls flat on there face.

Msg#: 1607 *US ROBOTICS*
02-29-92 12:25:00
From: TOM SMITH @ 930/1
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1465 (HST 16.8K ETC.)

David, HST_SALE's carried on the FIDONet Backbone and should be easily available to your or your SysOp. In fact, it's my understanding that both HST and HST_SALE must be carried if either's on the board. I may be wrong on this; I could be confusing them with the hard disk Echos.

If you're looking, I'd suggest that you also pick up MODEM_SALE and HS_MODEM. I've seen the former mentioned but have never seen a board I use carry it. I'm a regular reader and poster on the latter; it covers all high-speed modems. Good Readin'...

Tom Smith/Dallas...

Msg#: 1611 *US ROBOTICS*
03-01-92 00:33:00
From: JASON BUCHANAN
  To: ALL
Subj: DUAL STANDARD V.32BIS AND HST 16.8KBS

Hello!

I have been reading this echo with interest for several weeks with only sparse hopes that USR has released their Dual Standard modems with the HST 16.8Kbps mode.

Would someone kindly confirm whether USR has indeed started to ship Dual Standard V.32bis/HST 16.8Kbps modems, or if USR is waiting at a later date to announce them?

Many thanks in advance,
Jason Buchanan

Msg#: 1612 *US ROBOTICS*
02-28-92 23:09:00

From: DENNIS DOMAZET
  To: KLAAS HAMBOERGER
Subj: REPLY TO MSG# 1509 (HST UP-DATE TO 16.8)

* On 02-25-92, Klaas Hamboerger smashed keys to David Bernard about:
KH> think, that USRs behaviour isn't very fair. It was hard to
KH> save enough money to buy the modem. Now it is only one month
KH> old and already antiquated.

(I couldn't quote your whole message because it was too long, but I think that you remember what you wrote).

I agree totally with you. Let me narrate my own experience with USR.

Last October, I purchased a USR HST 14400 with v.42bis, the newest HST they had on the market. On the box that I purchased the modem in, it clearly stated, in two separate places, that the modem was upgradable to dual standard v.32 with "modules" that could be purchased from USR. That is one of the main reasons that I bought this modem, that the package stated that it could easily be upgraded.

Then I discovered that I needed v.32 capability, so I called and wrote USR about purchasing this module. They told me that it had been "discontinued". I found this fascinating, since the modem's chip dates were late August of 1991. I was very upset that USR would explicitly state on their package that the modem could be upgraded and then discontinue the upgrade so quickly.

Now, the only thing I can do is purchase another modem. I had faith that I would not need to spend a great deal of money to be able to use v.32, but I guess that I was wrong. I like the HST, I think it is a fantastic product, but USR's customer support leaves much to be desired. It really is unfortunate that something like this could happen. When I do buy a new modem, I regret to say that it will definitely NOT be a USR product again...

Msg#: 1655 *US ROBOTICS*
03-01-92 14:11:00
From: MIKE DRUMMOND
  To: LARRY NESBITT
Subj: REPLY TO MSG# 1491 (NEW 16.8K (OR WHATEVER) Q)

LNHH>   Like I said in the first message, I love my 14.4 hst modem and
LNHH>wouldn't change for the world! Now, I must admit that I encountered
LNHH>a problem with the service...but they stood by their product and sent
LNHH>me at no cost for service and return mail the finished product. Where
LNHH>I had a complaint was in shiping my modem to them....with insurance and
LNHH>mailing not to mention packaging....it cost me (after four times) over
LNHH>$90 dollars. This all could have been taken care of if the tech. would
LNHH>have noticed the loose screw on one of the connections. I know it doesn't
LNHH>sound very professional, but I can assure you that its still the very best
LNHH>modem on the market and I for one will continue to use nothing but US
LNHH>Robotics modems. Talk to you later and if your ZyXel modem is as good
LNHH>as you say, how come I haven't heard of them? But that is in another
LNHH>echo please...Or.....Net mail.

In the end all that really matter is that we are happy with the purchases we have made and all in all i am by no means an anti USR

advocate. As a matter of fact i usually recommend them to my users due
to the widespread use of HSTs. Although i must say it would sure be nice
if everything could talk to everything else.

Msg#: 1659 *US ROBOTICS*
03-01-92 07:23:00
From: TOM HENDRICKS
  To: KLAAS HAMBOERGER
Subj: REPLY TO MSG# 1612 (RE: HST UP-DATE TO 16.8)

 > behaviour isn't very fair. It was hard to save enough
 > money to buy the modem. Now it is only one month old and
 > already antiquated.

Although it still works every bit as much as before. Still provides the same
excellent performance, etc.

It is not antiquated. BTW: A new model will always be "On the way" at almost
any manufacturer I know of.

-Tom-

Msg#: 1663 *US ROBOTICS*
03-01-92 07:34:00
From: TOM HENDRICKS
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1659 (RE: HST UP-DATE TO 16.8)

 > previous modems. If the new DCE rate is
 > 16,800.. there is no guarentee or

This is a misuse of the term DCE (it means Data Communications Equipment, and
DTE means Data Terminal Equipment, and it specifies the wiring used in the
serial connection - has nothing to do with bps carrier rate.).

-Tom-

Msg#: 1682 *US ROBOTICS*
03-01-92 23:28:00
From: STEPHEN HENDRICKS
  To: HARDY ROSENKE
Subj: REPLY TO MSG# 1663 (HST UP-DATE TO 16.8)

 HR>   I was well aware of this, but I feel that you
 HR> are unnecessarily painting all dealers with the same

Possibly, but that is rapidly becoming the norm as the vast majority of
"computer dealers" in this area are closing and locking their doors to the mass
market. So a dealer is not very likely to support a product he did not sell.
That will depend on an individuals relationship with his dealer. A
relationship that is a good one will tend not to have the customer buying
direct from the maker. I used to work for such a dealer, and their doors are

now locked to the public.

```
SH> Unlike the normal customer your modem will be delivered out of USR
SH> Stocks in Chicago. This means that you will receive the most up to
 HR>        That is an interesting fact which I did not
 HR> know... I thought that they would be pushing out their
 HR> old inventory before the new, regardless of WHO or
 HR> WHAT the customer was.  Granted that USR is not set up
```

I won't assert that is a fact. But I believe it is normal business practise to ship down old stocks first. Any company today that is not on lifo accounting is in terrible danger. USRs customers are distributors. Distributors seldom have special needs for new "trick" modems. The distributors also buy is mass minimizing freight, labor and handling expenses.

```
SH> they save, and the fact that they are circumventing the normal support
SH> procedures! If you want better turnaround time on orders, then buy
SH> from your dealer, like NORMAL people.
 HR>        First: *_I_AM_NOT_NORMAL!!_*  Second: I do not like the tone that
```

No sysop is normal. We get special handling as the prima donnas we frequently are. Most are however just normal people who can't really afford the NORMAL price of a DS modem. You complain about the way of doing business associated with buying direct at a tremendous discount, that is a choice you have made. Normal buyers don't get such priviledges! You have the choice to buy through normal channels, and to get the better support associated with buying from an authorized dealer. Is it worth the extra $300 to be slightly inconvenienced?

```
 HR> is inferred in this, and I hope that you did not mean
```

There is no inference other than I don't think there is room to complain so much considering a nearly 50% discount that is attained soley on the basis of being a Sysop (and one that is apparently critical of the procedures involved in producing World Class Leading edge technology).

```
 HR> it the way that I am reading it....  Third:
 HR> "circumventing normal support procedures"???  Well,
 HR> let me just say that they had BETTER support my modem
```

They will obviously, but normal support procedures when I sell a modem involves immediate exchange for a new one, if there is a defect, and free shipping both ways, with me doing all the LD phone charges. Is that worth $300 to you?

```
 HR> just the same as anybody elses!!!  Hundreds of
 HR> dollars?  Yes, I am saving that, but mainly becuase I
 HR> am buying direct.... not paying shipping costs that a
 HR> DEALER would incur... I am
```

The dealer pays just as much as you do for freight (other than volume shipments). If you knew how much a dealer really made on a modem, you wouldn't wonder why so many of them don't provide support any longer. My former employee instructed me to charge no less than $70 per hour for support of any kind! Buying direct is a temendous burden on a manufacturer who has to hire a support staff to ship single units all over the country. Even if USR sold the modems to Sysops for $100 over production costs, they easily lose that much on the labor to process the order and put the modem in a shipping container and ship it. My former company told us (sales reps) that it cost no less than $30 to generate an invoice, and just watch the shipping people pack things like single modems. You can see that USR is being VERY good to Sysops for the positive PR they get from it.

```
 HR> also not paying a stocking fee or a warehousing fee and
 HR> the money I am paying
 HR> is not going into a salesman's pocket!  I worked in the
 HR> computer industry and
```

USR doesn't pay stocking fees or warehousing fees either. I think you are confusing the role of the distributors and that of the manufacturer. Even with $300 built in to the dealer, after paying shipping and two sets of handling and invoice charges, net 30 expenses, and possibly salaries and commissions a company wouldn't be making much money on an item.

```
 HR> a markup of 40-50% on certain products is the NORM!!  I
 HR> can safely say that I
 HR> am paying only slightly more for my modem direct from
 HR> USR than a dealer ordering a GROSS of them would
```

In the electronics industry a margin to list price of 40-50% is normal. The discussion here has been in relation to the lowest possible prices, not normal selling prices. If you purchased a DS from a dealer at a normal price of $975 then you would certainly be entitled to major support from that dealer. When we start talking in terms of giveaway prices from mail order houses, you get what you pay for.   It also sounds as if you have neglected at least one level of distribution, because the markups are not 40-50 in the real world at all.

```
 HR> pay....  Turnaround time, yes, I could go out and buy
 HR> a DS this afternoon for $800... or wait here and get
 HR> one shipped to me from USR for $550.....  I can wait.
```

The prudent thing to do. I just think we are lucky to have USR around!

Msg#: 1683 *US ROBOTICS*
03-01-92 23:50:00
From: STEPHEN HENDRICKS
  To: DAVID BERNARD
Subj: REPLY TO MSG# 1459 (HST 16.8K ETC)

```
 DB> |Currently USR HST modems have a VERY High resale value.  Judging from
 DB> |HST-Sale echo, the modems are in great demand.  If you have an older
 DB> +----------------------------------------------------------------------
 DB>  Hey Stephen,  Where is this confer available?  Maybe some special deals
 DB>  to come if 16.8 comes out & SYSOPs want to up-grade, but we don't get
 DB>  that confer around New Orleans & I have heard others asking too.
 DB>  Well thanks for the help.  Ltr....David
```

I'll see if I can find out for you. We have it here in Baltimore, but many areas don't carry all those confusing similar sounding echo names.

Msg#: 1688 *US ROBOTICS*
03-02-92 00:24:00
From: STEPHEN HENDRICKS
  To: MIKE DRUMMOND
Subj: REPLY TO MSG# 1577 (NEW 16.8K (OR WHATEVER))

```
 MD> formula the other half is support and this is where USR falls flat on
 MD> there face.
```

While I certainly understand how you feel, the course you have decided on is prudent. If however you do have a problems you can call on me for help with it as I am authorized as a USR dealer and will help anyone who needs it. I can not guarentee to do anything more than anyone else, except try. If what happened to you happened to me, I would have had it taken to the very top of USR management, quickly.

Msg#: 1724 *US ROBOTICS*
03-01-92 23:25:00
From: GEORGE PARDUE
  To: DENNIS DOMAZET
Subj: REPLY TO MSG# 1682 (RE: HST UP-DATE TO 16.8)

```
 -=> Quoting Dennis Domazet to Klaas Hamboerger <=-

 DD> Last October, I purchased a USR HST 14400 with v.42bis, the newest HST
 DD> they had on the market.  On the box that I purchased the modem in, it
 DD> clearly stated, in two separate places, that the modem was upgradable
 DD> to dual standard v.32 with "modules" that could be purchased from USR.
 DD> That is one of the main reasons that I bought this modem, that the
 DD> package stated that it could easily be upgraded.
 DD> Then I discovered that I needed v.32 capability, so I called and wrote
 DD> USR about purchasing this module.  They told me that it had been
 DD> "discontinued".  I found this fascinating, since the modem's chip
 DD> dates were late August of 1991.  I was very upset that USR would
 DD> explicitly state on their package that the modem could be upgraded and
 DD> then discontinue the upgrade so quickly.
 DD> Now, the only thing I can do is purchase another modem.  I had faith
 DD> that I would not need to spend a great deal of money to be able to use
 DD> v.32, but I guess that I was wrong.
```

Dennis, You might want to talk to your state Attorney General's office.
Or even Federal Attorney General's Office if you bought it mail order.
And the office of your Governor, and President Bush, and a few congressmen.
Also, follow up with letters to each of them, with a copy to the
president of the company that wronged you.

Hope you kept the box which has the upgrade info on it. Send a Xerox
copy of the message on the box also.

It's often amazing how companies can change their minds, "in light
of new facts which just came to our attention".

Talk atcha later,

George

Msg#: 1726 *US ROBOTICS*
03-02-92 10:42:00
From: TOM SMITH

  To: DENNIS DOMAZET
Subj: REPLY TO MSG# 1724 (RE: HST UP-DATE TO 16.8)

Dennis, did your HST with the upgradable labels on it come from USRobotics directly? If it came from a dealer, there's a good chance that it was sitting on his shelf for quite a while. If that's the case, then the wise thing to do would have been to either get a solid guarantee from him that the upgrade was available or check with USR on it yourself. If it came from USR, then I'd agree that you have a very legitimate reason to squawk, especially if you asked about the upgrade and was told by the USR sales rep that it was available. If it came from a dealer and you didn't take the needed steps to protect yourself, then all I can say is to remember the next time: Buyer Beware. There're plenty of warnings on nearly every piece of literature attached to a device which plainly state "Subject to Change" or some derivative of this. What it means is that a company can, and does, change its specifications on a regular basis. With this, I never ASSume that a device'll be the way it's advertised unless I check with the company and get some solid guarantees on the thing.

Now, for some suggestions on upgrading. First, if I remember, the HST USR upgraded for me cost something like $400. This's no great bargain, especially when you can now buy V.32bis-class machines for betweenn $3-400 which can include such bonus points as FAX and voice mail capabilities. You can easily sell your HST for enough to buy one of these puppies brand-new on the HST_SALE Echo. You can also buy one of them and a serial switch or port for less than it'd cost you to upgrade in the first place. Second, if you really want to upgrade, try dropping a wanted ad in HST_SALE, HS_MODEM, FOR_SALE, or CFOR_SALE. I used to say that it was impossible to find the boards, but someone reported very recently that he'd picked up one for about $135, so it appears that they're out there but hard to find. Good Luck in the Hunt...

Tom Smith/Dallas...

Msg#: 1740 *US ROBOTICS*
03-03-92 01:22:00
From: CRAIG SMITH
  To: GEORGE PARDUE
Subj: REPLY TO MSG# 1726 (HST UP-DATE TO 16.8)

On 03-01-92, George Pardue wrote to Dennis Domazet:

>..................................
Dennis, You might want to talk to your state Attorney General's office.
Or even Federal Attorney General's Office if you bought it mail order.
And the office of your Governor, and President Bush, and a few congressmen.
Also, follow up with letters to each of them, with a copy to the president of the company that wronged you.

Hope you kept the box which has the upgrade info on it. Send a Xerox copy of the message on the box also.

It's often amazing how companies can change their minds, "in light of new facts which just came to our attention".

Talk atcha later,

George

... Illegitemi Non Carborundum!

>...................................

True, truer, truest.That's some very good advice. I work with the law
on a daily basis and one thing that most states have is a Fair Trade
Practices Law and/or Advertising Law. In Texas, if you can prove unfair
trade practices and/or advertising, you are entitled to recoup 4 times
damaged and NOT pay for the item that you purchased. It's a long, winding
road through the legal system, but it does prove a point.

As Americans, our entire legal system stands on principles, not actions.
It doens't matter what they 'intended' to do, but what the 'General
Public' would have understood as the case. Sue'm, make'm give you the
Dual, or just have a good time watching your State Attorney's office have
some fun and games with a major corporation.

Craig

Msg#: 1755 *US ROBOTICS*
03-02-92 13:32:00
From: PAUL HALYUNG
  To: DENNIS DOMAZET
Subj: REPLY TO MSG# 1740 (HST UP-DATE TO 16.8)

 > Then I discovered that I needed v.32 capability, so I called and wrote
 > USR about purchasing this module. They told me that it had been
 > "discontinued". I found this fascinating, since the modem's chip dates

The Rockwell V.32 daughterboards are DEFINITELY not discontinued. Anyone who
says this is blatently a liar.

We have 2 brand-new GVC V.32/V.32bis V.42/V.42bis modems and after careful
inspection, contain the SAME daughterboard that can be found in an upgraded
older HST/DS modem.

The date of manufacture on the Daughterboard was 01/07/92. Go figure.

Same chips, same dual inline berg pins for the socket(s) in the HST etc.

Just need a new set of ROMs and away you go. I am going to persue this and not
let up. There is something drastically WRONG going on.

Paul

Msg#: 1763 *US ROBOTICS*
03-02-92 19:51:00
From: AL FILANDRO
  To: I GOT MY NEW MODEM!

Subj: COURIER HST DS 16.8K

Here is the ATI7 Info:
Configuration Profile...

```
Product type          US/Canada External
Options               HST,V32
Clock Freq            16.0Mhz
Eprom                 128k
Ram                   32k

Supervisor date       02/12/92
DSP date              02/06/92

Supervisor rev        4.1
DSP rev               11
```

Its small...but still looks like an hst..also since it is smaller, it appears heavier than the other hst I have here...Its kinda cute. It also supports a new connect rate of 16.8 "Connect 16800/HST/HST/V32bis" whatever...with one of its own kind. ---Contains V.54 for analog, digital and remote loopback testing--It also supports a DTE rate of 57.6k (aka lock your com port at that)...

Ill have to play with it for awhile but it looks real nice (even the DEMO tag on the top ..the other HST I bought didnt have that <G>) Thanks USR!

Msg#: 1778 *US ROBOTICS*
03-02-92 21:58:00
From: BRYAN HOLLEY
  To: DENNIS DOMAZET
Subj: REPLY TO MSG# 1755 (HST UP-DATE TO 16.8)

KH> think, that USRs behaviour isn't very fair. It was hard to
KH> save enough money to buy the modem. Now it is only one month
KH> old and already antiquated.

Believe me, they didn't plan to introduce a new modem just after you bought yours! Get real. Every company must continue to improve / enhance their products and they MUST be introduced at some time.

 DD> Last October, I purchased a USR HST 14400 with v.42bis, the
 DD> newest HST they had on the market. On the box that I purchased
 DD> the modem in, it clearly stated, in two separate places, that the
 DD> modem was upgradable to dual standard v.32 with "modules" that
 DD> could be purchased from USR. That is one of the main reasons
 DD> that I bought this modem, that the package stated that it could
 DD> easily be upgraded.

 DD> Then I discovered that I needed v.32 capability, so I called and
 DD> wrote USR about purchasing this module. They told me that it had
 DD> been "discontinued". I found this fascinating, since the modem's
 DD> chip dates were late August of 1991. I was very upset that USR
 DD> would explicitly state on their package that the modem could be
 DD> upgraded and then discontinue the upgrade so quickly.

Again, this is not USR's fault. Their supplier, Rockwell, has discontinued the

V.32 module and gave the companies that used it no alternative other than to use the newer V.32bis module. Unfortunately, it is not totally compatible with the older V.32 module.

Msg#: 1798 *US ROBOTICS*
03-03-92 17:43:00
From: MIKE DRUMMOND
  To: STEPHEN HENDRICKS
Subj: REPLY TO MSG# 1688 (NEW 16.8K (OR WHATEVER))

SHHH> MD> formula the other half is support and this is where USR falls flat on
SHHH> MD> there face.

SHHH>While I certainly understand how you feel, the course you have decided on
SHHH>prudent. If however you do have a problems you can call on me for help wi
SHHH>it as I am authorized as a USR dealer and will help anyone who needs it.
SHHH>can not guarentee to do anything more than anyone else, except try. If wha
SHHH>happened to you happened to me, I would have had it taken to the very top
SHHH>USR management, quickly.

Well thanks Steve. I must say that your attitude is much more "consumer friendly" than the service department. You must do a fairly good business....

Msg#: 1933 *US ROBOTICS*
03-04-92 10:34:00
From: TOM SMITH
  To: PAUL HALYUNG
Subj: REPLY TO MSG# 1778 (RE: HST UP-DATE TO 16.8)

Paul, USRobotics has dropped marketing of the Rockwell daughterboard. Since you need a set of the properly-encoded PROMs to make it work, and since you can legally only get a set from USR, this effectively means that, so far as the USR world goes, the Rockwell card has been discontinued. I've seen one message from a person who found one on the HST_SALE Echo, but I've also seen literally dozens of messages from people looking for them. While you may be technically right in that Rockwell's still building the boards, if USR's chosen to not sell and support them then the assertion that they're no longer available in the Courier line's correct...

Msg#: 1948 *US ROBOTICS*
03-03-92 07:59:00
From: STEPHEN HENDRICKS
  To: PAUL HALYUNG
Subj: REPLY TO MSG# 1933 (HST UP-DATE TO 16.8)

 PH> Just need a new set of ROMs and away you go. I am going to
 PH> persue this and not let up. There is something drastically
 PH> WRONG going on.

Pursue it outside of this venue. After you get your answer let us know.

Msg#: 1958 *US ROBOTICS*
03-05-92 02:28:00
From: CHRIS PRATER
  To: GREG GORE
Subj: REPLY TO MSG# 1683 (HST 16.8K ETC)

 > Hi Manuel, that is just a rumor. There is no such
 > thing. If I am not
 > mistaken I bieleve the fastest their is V.32bis which
 > is 14 400 bps. The
 > next step up that USR is developing is called V.FAST
 > which won't be out
 > for a couple of years and it will operate at 19.2K!

The 16.8HST/v32bis14.4 is no rumor. I've called 1-800-DIAL-USR and they confirmed it. When you order a new modem now, it will be one of these V.small modems.

Plus a few people in this echo have said that the new modem they've received have been the new, smaller, faster, v.small modem.

:)

/\Chris/\

Msg#: 2168 *US ROBOTICS*
03-11-92 07:37:00
From: TOM HENDRICKS
  To: RAY MANN
Subj: REPLY TO MSG# 1948 (RE: HST UP-DATE TO 16.8)

 > Just wondering: if US Robotics doen't have any Rockwell
 > daughterboards left, how do they repair broken
 > Dual Standard
 > modems that have a Rockwell daughterboard? Hmmmm....

Why don't you call them up and ask them?

It's better than discussing it here, without any knowledge on the subject whatsoever.

-Tom-

==

END OF MESSAGE CAPTURE!!

==

SPECIAL THANX GOES TO: FREEJACK, NAVIGATOR, ZELNIK, XTC, AXE, CAPRICORN, AMOK, NOMAD, TOM-CAT, MYMURTH, THARGOID, DEATHLOK SPIRIT & SHARK!

```
GREETZ TO: NEMESiS (Yeah BOYZ!), CRYSTAL (What's up with that Nosferatu?),
           QUARTEX (Whats up?), & ANTHROX, and all the other groups that
           think they deserve a greet! :)

         If ya like this text file call D'YER MAK'ER AT: +908-730-8633

               _________  __  ___     ___  ______________________
               \|||||||||\ \/ /|||\__/|||\/||||||||||||||||||||||\
               /''''''''''\   \''''''''''/''''''''''''''''''''''/
              /    /~\    \   \___     /     ~~~~~~/    ~~~    /
             /    /__/    /______/    /    _______/    _    _/
            /            /           /           /    / \    \
            \___________/\__________/\__________/\___/   \___/

     ______________  ____________  ____    ____  __  ______________________
     \||||||||||||||\/||||||||||||\/|||\  /|||\ \/ /|||||||||||||||||||||||\
     /''''''''''''''/''''''''''''/''''/_/''''/   /''''''''''''''''''''''/
    /    /   /    /    ~~~     /           /   /     ~~~~~~/    ~~~    /
   /    /\__/    /    ___     /     _    _/   /     _______/    _    _/
  /    /   /    /    /  /    /     / \    \  /            /    / \    \
  \___/    \___/\___/   \___/\___/   \___/  \__________/\___/   \___/

     _____  _____  __  __  _____  _____  __  _____        _   _  _____
    /   | \/   __>/  \/  \/   __>/   __>/  \/   __>      / ]_[ \/  _  \
   /    |  \   ___\       \   ___\___   \   \___   \    /        \ / \  \
   \  |    /       /      /       /     /  /       /    \   _    / \_/  /
    \_|___/\_____/\__\/__/\_____/\_____/\__/\_____/      \_] [_/\___\__>

                __   __   __     ___ __   __     __      __  __
          | __ (__) |  | (__) __   / __) |  | __ (__)  /_  __) __)
          |       /  |__| (__)    /  __) |__|    (__) (__) __) __)
------------------------------- END OF PHILE! ----------------------------
      _______________________  /\_____________________________
     /                   /   \/  \           /     \       / /\[SP]/\
    /                   /         \         /       \     /\/  \  /  \
   /     ______________/   _      /        /__      /    /  \   \/    \
  /     /___/    ___/_    |/     //   |___/__/     /    /    \         \
  \    /    \        \   /      //    ||    /     /\   /    /           \
   \         \       /  /      //     ||    \       \      /           /
    \________/\______/__/     / \_____||____|\_______\______/__/\  /\___/
                     \____/                                     \/

                      3 1 2 - 7 7 6 - 0 4 1 7

            UPLOADED ON 14-Mar-92       TIME UPLOADED 12:02:29
```

==
DOCUMENT vidomac.app
==

TRANSCRIPT OF SEMINAR:
"Macintosh in Film and TV Production"
MacWorld Expo, San Fransisco, January 17, 1986
(Edited for Clarity and Brevity)

PARTICIPANTS:
ARTHUR GREENWALD, Moderator, Creative Services Director, KDKA-TV/Pittsburgh
RICHARD HART, Co-Host "Evening Magazine", KPIX-TV/San Fransisco
STEVE KOTTON, Co-Owner, Pacific Video Resources/San Fransisco
ANTHONY REVEAUX, Media Critic, Lecturer in Film History, Sonoma State Univ.

GREENWALD: This session came about largely out of my own frustration in calling everybody I could think of at Apple to get some support in marketing the Macintosh to our industry, film, television production, advertising... It seems to me obvious that a graphics oriented machine like the Macintosh has obvious advantages for graphics-oriented industries like ours, but those advantages aren't always obvious to our employers. In commercial television especially where the business side is very often separated from the production or creative side. Business decisions such as bulk purchasing or compatability don't always have a lot to do with how computers are really used on the job.

Today we'd like to touch on today is the success we've had in using the Macintosh in our own work, and we've also invited here today some developers of specialty products for our industry. They'll get a chance to say a few words about their product.

I'll begin by describing how I use my Macintosh in local television production. I was first attracted to Mac for its graphic potential, but I became a junior Mac Evangelist because it's so easy to use that it occured to me that with the high turnover rate of employees in TV stations and ad agencies that this was a pretty vital characteristic, too. You can actually train a short term employee in a day to actually use the machine. And then because it's so easy to use, it's self-reinforcing, people continue to use it.

I immediately started using MacPaint to design print ads and simple storyboards for dramatic scenes and for the simple animation our station producuces. It was invaluable a communications tool. I could take my description of how our logo should move or shimmer or so forth, take that to the station manager and show the proposed animation step by step. Like any storyboard it gave us a common means of discussion, but it was much easier to revise. Since then I've come to use Hayes' Smartcom II software with a Hayes modem which lets you use one picture at a time from the Scrapbook and show it and change it in real time over phone lines. So that's a real godsend to be able to talk to an animator in Los Angles while I sit at my desk in Pittsburgh. That can eliminate unnecessary and costly trips to the west coast.

Word processing. I prefer Microsoft Word if only because it can open more windows that MacWrite. Just that ability to change type sizes, which we take for granted as Mac owners, well that's a real advantage when you're trying to indicate the relative size of supers or text in a print ad. I've abandoned our usual art order form because the output of the Macintosh shows what I'm looking for much more clearly.

One general observation is that some of the specialty software that I THOUGHT would be terrifically useful--- such as Videoworks and Slideshow Magician, to name two I admire a great deal--- I haven't found much opportunity to use those in my work. We have too many deadlines in local TV for me to take the time to use that software to make polished presentations. Perhaps those of you in advertising who can take more time with each job have found those more useful. I find MacPaint and MacDraw do more or less what I need to do.

Finally, since acquiring a modem, I've gotten very involved in telecom-municating. I'm particularly active with CompuServe. I'm an Associate Sysop now for the Broadcast Professionals Forum on CompuServe, which is an exciting new way for us to share ideas and opinions about our industry instantaneously. It's also a good way to upload specific problems or questions to the board and come back a few hours later and get some good professional replies, not merely technical tips, but creative ideas on lighting, promotion, casting and more. Plus a variety of freelancers have begun to upload descriptions of their services and where they work, etc. It could very well become a new way of networking freelance work.

Now Steve Kotton will describe how the Macintosh and Lisa have been useful to his independant video work.

KOTTON: Yes, I was one of those fortunate or unfortunate souls who got into icons very early. I was a little disappointed with Apple's response to it, but I'm very excited to see the kinds of hardware and applications that are here today.

I run a small facility here in San Fransisco, Pacific Video Resources, we function both as a facility, we have three complete edit rooms, but we also do full productions, documentaries and other programming that's on from the commercial networks to syndication, cable, all over the place. I'll just run through some of the software we have and use on a day to day basis and why the Mac has become so essential for a creative small business.

First, I'd like to agree with Arthur about how easy it is to use, and to train freelancers... in twenty minutes. They can start real work for you almost immediately. I have colleagues who have owned IBM's and they still don't use them.

For scheduling edit rooms and production equipment, Front Desk is a wonderful piece of software. It can schedule different times, months, plus it does reporting functions for billing. Check it out, it's really a very good program.

Overvue. We checked out about ten databases. We're using Overvue for a couple of specialized functions. For equipment rosters, serial numbers, for insurance companies, and to log maintenance.

MacDraft. We have our entire facility diagrammed. All the special equipment we've made up is all totally documented on MacDraft. Being able to just pop in a disk and just check out an area where a wire may be bad, for engineering it's just amazing. Our three edit suites were designed on MacDraft. I designed a production truck this summer using MacDraft. It's really an amazing tool.

MacPaint is wonderful for storyboards, especially for effects work where you can get into detail and show how that effect is going to look and when and where it takes place on the screen. I have used Videoworks for a certain bit of animation and while it is more tedious than just MacPaint it certainly is a nice little package.

Using Excel. I find Excel to be one of the best spreadsheets that I've seen coming down the pike. I put my form of the AICP bidding form into Excel and it's really a great spreadsheet for that. Glenn Przyborski in Pittsburgh has placed the entire bidding form into Excel and it's extremely useful. It's great that you can get on the phone and within 5 minutes have at least a good start of a bid that used to take hours and hours to do.

There's another program you out to check out it's called Document Modeler by the Model Office company. If you do a lot of correspondance which we do when we're doing bids or talking to clients, it's sort of a form letter generator but much more personalized. You can input a number of different responses and then pick and choose among them to fit the job that you've got. It really puts out a letter that is very personal and yet is a form letter that gets those responses out quickly to clients.

Finally, we also use Pagemaker a lot. We try to do our own publicity in house and we do a newsletter once a month, all on the Mac, all on Pagemaker and the Laserprinter. We also use MacDraw and the Laserprinter to design shooting schedules, editing forms, logging forms, character generator forms...

The Macintosh still has a ways to go in terms of specific pieces of software for our needs, but it's still far ahead of any other computer out there. With its graphics capabilities and the variety of software, it's really ideal for our industry.

HART: The show I do here in San Fransisco on KPIX, Channel 5 is Evening Magazine. In most markets around the country it's called PM Magazine. The distinction is that those stations owned by Westinghouse Broadcasting call the show "Evening" and anybody who buys the show from us calls it "PM." We shoot 100% of our show on location. We shoot nothing in the studio.Our kind of work is different from what a television newsroom might do.   I worked in the first broadcast newsroom -- radio or TV -- that was computerized. That was KCBS, the CBS-owned radio station here in San Fransisco. It's about 11 years ago that they first brought in terminals. That, of course, met great resistance from the old-time reporters at 'CBS who had covered Pearl Harbor. Their favorite was the old Olympia manual typewriter.   And they scurried and hid them away under their desks so when they had to do "news" they'd haul out the Olympias. This is true!

The NEW hot setup is one designed by a guy who used to work for Colorgraphics. Imagine a guy working on a live newscast for radio who wants to constantly monitor Associated Press, United Press for bulletins. Now those services code their stuff "Level 1..2..3" alerts. Audio feeds, too. It would be nice if you were delivering a newscast and on the radio or something and sudddenly the corner of your screen would flash and alert you to a "Level 1" situation, you'd hit a key combination and be reading what's available.

The guy who left Colorgraphics has developed a very Mac-like system now. But he's not allowed to compete with his old company for another two years in this country so he can only do it in Australia, Japan, and in some countries in Europe. And I'm on my way to see it next week, but they tell me it uses a mouse and the whole system such as we dreamed of ten years ago, very well. He's done this on an IBM PC system and he's having a lot of problems with resolution because among other things, he uses it for editing tape too. He has a little image on the screen of two reels. When you're splicing audio tape, the tape is literally spliced.   The system uses speech digitization that is so good that you can actually edit audio on the screen with the mouse. I'm convinced there IS a way to do all of that on the Macintosh. He began in the IBM world, and he strictly used IBM terminals, I don't think he's explored the Macintosh. I'm going to talk to him next week about that, to see whether his company wants to do something of that nature on Macintosh.

The ideal newsroom situation would be to read right off the screen the entire newscast and as the news changed or new news came in, instead of someone handing you copy, it would be scrolling on the prompter off of a computer screen. Nobody's doing that yet. It's possible now, but everyone's afraid to take the first step just as they were with the rest of the equipment. When it comes to using electronic equipment for typing news, our newsroom at KPIX is as backward as any in the country. They still type manually and scroll taped sheets of paper through the machine. (SYMPATHETIC LAUGHTER) I mean, it's 1986 and my station is still hand typing with the big typewriters that have the big letters on them. And the last two news directors have this GREAT reason why they haven't switched over: "Well, we're waiting for the price to come down." (LAUGHTER) "Or until they build a better system." So figure by the year 2012 we ought to get electronics in there.

Typically what we at Evening Magazine do in a day is shoot a daily half-hour show which is divided into 3 or 4 feature stories, each of which is scripted and edited -- then the introductions, the "Good evenings," etc. which are wrapped around that. Obviously we do a lot of writing for the show, but not on a deadline basis as the newsroom does. If we want we can do our typing in the field.

Typically, if we shoot a story -- say a 5 minute feature that's going to air in 2 weeks (We shoot about 30:1, about 30 minutes of tape for every 1 minute of story)-- there's a producer charged with pre-editing that story, doing a cut sheet (edit plan) for the editor, which contains the incues and outcues of cuts he wants to use from the interview. It also has the voiceover script for me or my co-host to record. Basically it's a sheet of paper that maps out the order of all the pieces of audio and video that make up the story. This process may take two or three days so we have the opportunity to trade ideas.

A lot of conferencing and changes take place before video editing. Usually that means a lot of pencil editing, but obviously it's better and easier to make those changes electronically, on disk, or better yet, by leaving drafts for each other on a system like CompuServe. (Incidentally, although Art and I work for the same company, we MET on CompuServe.) For the past year, several of the producers and I do just that. One of the producers will upload his script to CompuServe. Then at my leisure the next day at home or even at my desk at work, I can download his script. I can edit it electronically and if he's happy with my changes, either of us can print it out to be recorded in the booth.

We do this for about two or three scripts a week. The nice part about storing it on CompuServe is we don't have to both be online at the same time. We travel a lot and this system allows us to download scripts anywhere there's a phone. If I have to re-record a line while I'm out of town, I'll sometimes record the new script onto a videocassette in a hotel room or wherever, and ship it back by air.

The next area is graphics. Now a Macintosh graphic can be uploaded for me to download so that the editor can get an idea of how the pictures should go together. Now a cut sheet with incues and outcues is nice but we can actually give an idea of how the picture flow ought to go in the piece. What we're aiming for is for the producers to upload a kind of storyboard to guide me and the editors. I think the Macintosh is the only thing that will allow us to do that kind of thing efficiently and on a regular basis.

The funny thing is that KPIX has about 300 employees and all the Macintoshes are coming in the back door. Because the official word from our computer headquarters on the east coast is the company will support only certain Burroughs and IBM equipment. So that's all we can buy. Some people have hidden Macs in their operating budget instead of their capital budget and other tricks.

There'a guy at our station responsible for commercial production who's been experimenting with Concertware and many other programs trying to find one to provide musical accompaniment for the jingles and commercials produced at KPIX. There's a freelancer who will do a complete transcript of a videotaped interview for a producer and put them on a disk. So when our producer writes the script, he can for instance, in Word, put up two windows. In one display the actual transcript of what was shot on tape and in the other window write his voiceovers and how it will be cut together. There are some other uses which are more esoteric, but that's the basics of how we're using the Mac right now.

REVEAUX: We've talked about film and television. I also work in multi- image slide prodution. That's an area where the Mac's pixels are only being scratched, but which has a lot of application to film and TV. Right now it's only terms of doing scripts. When I did the cover story for Macworld, I made a list of all the ways people had scrounged trying to come up with a way to process two-columns of text for scripts, even in MacProject. We don't really have that ability yet. What we're going to need eventually is some sort of integrated script format that chains your two columns together shot by shot. So that even 30 pages in if you make a change in a shot, it will always keep the shot number, the sound and picture, chained together. I hope we see that in our lifetimes.

One nice thing is when you're doing scripts for clients is that with MacPaint and Clip Art you can have a nice big copy of your client's logo on your cover page. The library of Clip Art expandeth as we speak. Right now I've been doing more slide shows in terms of projection for performance in the art world. Opera, theatre, dance. Right now I'm working on a full- length avant garde opera By David Ahlstrom the San Fransisco composer based on the writings of e.e. cummings. And for the first time now, instead of doing it all photographically, I'm doing it mostly on the Mac. And here's one thing I've found, to get this kind of vivid neon look, of letters or pictures, you bring it in there and then just select Invert. Then put a colored gel in front of your camera lens of whatever color your want those lines to be. You have no idea what I went through to achieve that same effect photographically. You have to take into account the blue cast of the Mac's tube.

Some of the most exciting new technical developments for using the Mac in our industry are the audio digitizers that are now available. Just as video digitizers like MacVision and Thunderscan can transfer external images into MacPaint, you can now do the same thing with sound. You can digitize a sound in a manner similar to the high-end machines like the Kurzweil or Mirage (they cost tens of thousands of dollars.) The Kette Group, The MacNifty people, offer a low-priced digitizer called the Sound Cap. It includes some clever "goodies" including an eerie one called TypeWriter. It mimics the sound of an old Smith Corona manual as you type on your Mac.   Now you can have a sound effect or a voice or music in short files, limited only by memory.

There's a new utility now in development called Sound to Video which allows you to put these sounds into VideoWorks. It's adds sound effects, or your own voice. I mean, Macintalk is nice but it speaks in "droid."

Magnum is about to release Slide Show Magician 1.3 which is really excellent. Not only does it have sequencing but also cinematic wipes, which in multi-imaging you'd need at least a six projector show to do that convincingly. With THEIR sound digitizer called Natural Sound, you can then hook these sounds into Slide Show Magician. VideoWorks can do a splendid slide show also. In fact, with Slide Show Magician and the sound program, you can have it actuate a tape deck, audio or VCR, OR, you can have the tape deck trigger the Mac. It's also coming with a couple of disks of digitized sound effects. I think of it as Clip Art for the ears. I'm sure we'll be seeing developers coming out with "albums" of sounds from nature, space sounds, etc. Some sound files are already available on CompuServe. What's more Slide Show Magician incorporates Macintalk. More and more Mac programs are coming out with digitized speech and sounds.

A few other things that our here... Graphics Magician by Penguin Polarity Software, no better or worse than Ann Arbor's animation program. The main thing is that it has full programmability. If you know Basic or C or Pascal, it gives you the program hooks to put animation sequences in your program.

Another animation program coming out is MacMovies by BechTech which is full screen 30 frame per second animation to be released in about 2 months, to be used with the Chromatron Color System.

Also Easy 3-D really IS easy, I've used it. You really can create shaded solid models within reason.

Also coming up is ComicsWorks by Mike Saenz who did SHATTER. Let me tell you, that is going to be one of the hottest things and here's why. You strip away the bug-eyed monsters and rocket ships that Mike has so carefully drawn there and it's one of the best programs for quickly mixing graphics and text that I've ever seen. It allows word processing in captions and balloons. It's ideal for storyboards. This industry is really so funny. Here we have this marvellous program. Now if he called it "Business Comic Works" then it would be respectable (LAUGHTER.) It's due out mid-April by Mindscape.

There are a lot of real sleepers out there that maybe we can use in our work. One of them is Fontastic, by Aldus, a wonderful font editor. If you do nothing else from Fontastic but switch things around from fonts, you can customize a font with a lighting grid or camera position markers, you can actually "type" into MacPaint diagrams of dials, lighting grids, etc. It's wonderful for training purposes. Arthur?

GREENWALD: Thanks, Tony. In a moment, we'll hear from some of the developers of specialty hardware or software for our industry, but first a word about finding software that will let us process text in columns. It's true that it doesn't exist. I've even resorted to using MacDraw which at least lets you put the text for a short script in columns, but with no word processing ability. But the people from Microsoft, who produce Word, are sympathetic to the problem and have said that if enough people write, they will very seriously consider implementing that in a future version. In fact at one time it was planned as a Word feature. The person you can write to, if you'll please join my letter writing campaign, is Mary Batterson, Public Relations Supervisor, MICROSOFT, 10700 Northup Way, Box 97200, Bellevue, WA 98009.

I mentioned before that you could write to me, and send me a blank disk, and I'll duplicate onto it the various software templates we're collecting for film and TV producers. Send the disk to Arthur Greenwald, KDKA-TV, One Gate Gateway Center, Pittsburgh, PA 15222.

So if the developers would now raise your hands, we'll invite you up one at a time.

MAN: I'm from Stanford University and we've developed a blocking simulation for the theatre students. We hadn't really thought about it in terms of film when we started, but some people have expressed interest in using it. We've developed an interface which the students can learn in 15 minutes and block a scene in about 2 to 3 hours. You can have the characters turn--their heads turn independant of the body-- you can have them standing up sitting down, lying or kneeling. We picked these as major body positions that represent life.

REVEAUX: I think you're being much too modest about this.

GREENWALD: I agree.

HART: This is my favorite program of the entire show here. Some of you have seen it. It's in the University Consortium corner. This is what impressed me about it. If you're blocking out a scene, you've got a library --- is it a library yet or is it a MacPaint document?

MAN: It's a library (of backgrounds) but any MacPaint document is a stage.

HART: Shakespeare said that (LAUGHTER).You've three elements, you've got characters, you've got movements on the stage, and the stage. The amazing thing is you've got a stage you can make in MacPaint then a menu of characters. Maidens, uh...

AUDIENCE: Swains!

HART: Thank you! Swains, swainettes.   If you want to populate your stage with characters you click on them. And you not only click on them as designated players, but you can click on a subcategory of "Extras" then from that menu you can choose potted plants and balconies and things. (LAUGHTER) I'm serious, and you can plan out the entire scene.

GREENWALD: In short, if you haven't seen it, you owe it to yourself. It's called The Theatre Game.

STEVE GREENFIELD: I'm Steve Greenfield from Screenplay Systems. We've developed something called Scriptor. We've just released the Macintosh version with a full Mac interface. And it's actually a little bit more powerful than the our IBM version. We're also the developers of a program called Movie Magic which is a budgeting, schedule and breakdown program for the IBM PC. We hope that it will be available by late Spring.

Scriptor is for writing features, TV movies, and one hour dramatic shows, and shortly, theatre. We don't deal with left side, right side, but I can tell you the people from Microsoft are more than just listening, give them a chance and they'll probably come up with something you'll like.

STEVE BECK: (of Beck Tech) I'm the guy who made page 73 of Macworld this month where they're showing our color Macintosh. I know in a room like this I can address video and television professionals who can appreciate not only are we getting color from the Macintosh, but we're converting to an NTSC broadcast standard signal. It's fully interlaid, fully equalized, all the widgets that let you take the signal from your Mac and mix it in with your production. So some of these products you've been describing effect what goes on BEHIND the screen but with our Chromatron, everything you see on the Mac is converted in real time to video. We also have a genlock overlay module coming out so you'll be able to genlock the Mac onto a videotape playback and then overlay Macintosh (key) graphics.

The other product we have is our MacMovies software animation package and it's a little different from a program like VideoWorks because this program does in fact let you playback full screens of Macintosh displays at rates of up to 30 frames per second. We have a demonstration of Olivia Newton John singing on the Mac. At Siggraph people walked up and said, "Oh I didn't know the Macintosh had gray scale" or "What'd you do, put a little television set inside there?" No, what we've done is develop a tool kit for working with images on the Mac that is sort of like the Basic language. We have a picture interpreter so you can build a little movie with MacPaint or MacVision documents and see it run as you build it with the interpreter. Then when you get the movie the way you want it, you compile it with the Movie Compiler. Then you can put it on a release disc with a program called the Projector.

Now all of this relies on a compression technique where we can squeeze as much as four megabytes of pictures down to five to seven hundred kilobytes and play them back. So with our 1 mg in the new Mac Plus or with our 2.5 Mg upgrade it's possible to put a full 30 second length movie in the Mac and play it back at full speed with color. So you're talking about roughly a 5 to 6000 dollar desktop color video animation tool based on the Mac and we think that's very important.

GREENWALD: Those of us who've had to worry how to find the budget money for a $125,000 color graphics machine can appreciate the fact that something even EXISTS in the 5 to $7,000 range. It's nice to hear.

JOHN WEYGANDT: I'm John Weygandt, college professor of theatre design at Pomona College in Clairmont, California. I'm using Business Filevision in my lighting design work. I find it amazing that Business Filevision thinks EXACTLY the way a lighting deisgner works. It makes a ground plan view of all the lighting instrument symbols, and then underneath that view, stores pertinent data. You can then pull out that data to make all sorts of lists about it: gel cutting schedule, dimmer hookup, instrument schedule, all that kind of stuff.

That's exactly how Business Filevision works and I've developed a template that uses symbols. I've created a font called "Blocks" that has 125 different lights so that the light can be a front light, backlight, sidelight from either side. And just paste it right into the document, then format your gels, dimmers, all that stuff. For example, probably the most amazing one is my gel cutting schedule. When it's time to cut gels, it'll start with the lowest number, say, a Roscoe Lux 04. And it'll tell me the location: "Electric Number 1" and then say "Instrument Type: 6" Ellipsoidal" 5 cuts, then a 6 by 12 ellipsoidal, 6 by 16, etc. And it'll total all those cuts at THAT LOCATION. Then it'll go on to the next location, say, "Electric Number 2" and then it'll go on to Roscoe Lux 05. So it's a great tool.

JODY BARAM: I've created the Video Production Planner System. I've taken several different modules, a staff and equipment module to track your people and equipment.   You can track them on a map or however you'd like to. I've also got a production module where you create electronic storyboards. And I just want to say that I can (inaudible) in columns.

I also have a Scheduler, a Studio Production Board, and also a live studio work scheduler including a calendar to keep track of all the activities and your coworkers. And I also have an edit lister which will keep track of shots to be edited and you can use that along with your storyboarding module to keep track of specific shots.

GREENWALD: Indcidentally, one product that's useful but certainly not as elaborate as Jody's template is Daykeeper by Dreams of the Phoenix. It's a simple appointment calendar that can be easily modified to track your production schedule. It allows you to assign priorities. If you need a simple deadline list tied to a calendar, I've found that to be easy to update.

MICHAEL EDWARDS: I'm Michael Edwards and I've just released a line animation system called DYNAMO. Most of you are familiar with VideoWorks where you build a picture by MacPaint and build a number of these pictures and display them rapidly. This is how television works. Another way of doing it is to allow the entry of a structured piece of information with that picture and another structured picture, and then perform a mathematical interpolation to aid in the smooth transformation from one picture to the next. In real time so you get smooth motion.   This reduces a lot of the work required because you only have to enter the initial data and not the later changes.

By incorporating a structure inside the program, you basically build structures representing the body. So you want to move the upper torso for example, you move the chest and the whole upper body moves with it because it's all tied together mathematically. The product is a simple line drawing system that allows you to enter thousands of frames depending on the size of the memory and allows enter line drawings. It's a shareware product, and it's getting up slowly on the various bulletin boards.   You can also buy a registered version.

MAN: I'm representing a friend from ABC Software. What he's come up with is a disk for MacPaint documents. 27 production forms basically just to provide well-designed breakdown sheets, casting information, commercial call sheets, daily production reports, deal memos, group releases, independant contractor invoices, petty cash, storyboards, minor releases, and much more. It's called Mac Movie Forms and all of them can be modified in MacPaint.

DANIEL SABSAY: My name is Daniel Sabsay and I'm a software engineer. I'm about to release a program called MacPrompter, which allows you to use the Mac itself or an external monitor as a teleprompter. We'll be increasing the product eventually so it can network and the display can be controlled, and the text edited, from another Macintosh. Right now it will only handle ASCII files.   However, you have the ability to drop right into the middle of the document somewhere with the selection menus provided. So if you're speaking in an interactive way, and you're asked a question, you can jump to a portion of the prepared text that answers the question.

There are several other features. You can adjust the scrolling speed as you read, and even record minute speed changes as you rehearse. MacPrompter will play back the text with all the same speed changes. You can go back through and modify any section of the script as you go.

I'd also like to mention a product by a company called Comtrex has a camera for

$480.00. It's a very high resolution monchrome video camera. And it can look at any part of a Mac screen and it synchs automatically to the Mac's frame rate so you don't get a roll. And you point the camera at the Mac and it gives an NTSC video output with beautiful quality. It cleans up the signal. A marvelous little gadget.

GREENWALD: (Repeats address for free disk) Please put your name and address on your disk label as well as your envelope. We're going to set up some Macs now to demonstrate some of the products you've just heard about. This ends the formal part of our presentation. I'd like to thank my fellow panelists for sharing their expertise. Thanks also to the developers who took time to be with us today.   And of course, thanks to all of you.

```
=============================================================================
DOCUMENT vt100
=============================================================================

Typed Up By Landon Statis / ProAlt Networks

DEC VT-100 Compatible Cursor Command Sequences

-----------------------------------------------------------------------------

----------------------------
Cursor Positioning Sequences
----------------------------

Name  Sequence     Default   Description
----  --------     -------   -----------------------------------------------

CUU   ESC[PnA         1      Cursor Up Sequence: moves the cursor up Pn
                             lines at same column.  Cursor stops at the top
                             margin.

CUD   ESC[PnB         1      Cursor Down Sequence: moves the cursor down Pn
                             lines at same column.  Cursor stops at the bottom
                             margin.

CUF   ESC[PnC         1      Cursor Forward Sequence: moves the cursor right
                             Pn columns in the current line.  The cursor stops
                             at the right margin.

CUB   ESC[PnD         1      Cursor Backward Sequence: moves the cursor left
                             Pn columns in the current line.  The cursor stops
                             at the left margin.

CUP   ESC[Pl;PcH  Pl=1,Pc=1  Cursor Position: moves the cursor to line Pl,
                             column Pc.  If either the default values or 0 are
                             selected for Pl or Pc, the cursor moves to the
                             first line or the first column.

                             The numbering of lines and the ability to move
                             the cursor beyond the margins depends on the
                             origin mode selection.

HVP   ESC[Pl;Pcf  Pl=1,Pc=1  Horizontal And Vertical Position: this sequence
                             operates the same as the cursor position sequence
                             (CUP).

IND   ESCD          None     Index: mov/es the cirsor down one line in the
                             same column.  If the cursor is at the bottom
                             margin a scroll up is performed unless the screen
                             lock mode is set.  In this case, the index
                             sequence is ignored.

RI    ESCM           1       Reverse Index: moves the cursor up one line in
                             the same column.  If the cursor is at the top
                             margin, a scroll down is performed unless the
                             screen lock mode is set.  In this case, the
                             reverse index sequence is ingnored.
```

```
NEL   ESCE         None      Next Line: moves the cirsor down to the first
                             column on the next line.  If the cursor is at the
                             bottom margin, a scroll up is performed unless
                             the screen lock mode ius set.  In this case, the
                             cursor is moved to the first column on the bottom
                             line and the scroll up is not performed.

SC    ESC7         None      Save Cursor: saves the current cursor position,
                             graphics rendition (screen attributes) and
                             character set selection.

RC    ESC8         None      Restore cursor: restores the previously saved
                             cursor position, graphics rendition, and
                             set selection.  If none were saved, the cursor
                             moves to the origin.

-----------------
Erasing Sequences
-----------------

Name  Sequence     Default   Description
----  --------     -------   -----------------------------------------------
```

```
==============================================================================
DOCUMENT wings.fury.cht
==============================================================================

 Brd ->General Information and stuff
Numb ->19 of 22
 Sub ->hey
  To ->all
From ->Mr. Substance (#38)
Date ->04/24/89  07:32:00 PM

Im new from Wa. Just uploaded a pic i drew a while back called Thexder. Its an
original freehand drawing of the box color but looks better...(i modified the
background to my taste (new order)) so, hope ya check it out and enjoy it.
Also, here is a cheat for Wings Of Fury
 Access the monitor after booting and selecting a level
 for infinite bombs, missiles, and torpedoes, type these three things
 01/a9aa:ff
 01/a9ce:ff
 01/a9bd:ff

And there ya have it..
Mr. Substance
```

==
DOCUMENT wizardry.4.info
==

<From>: Data Man
<Date>: SUN JAN 3 7:31:00 PM

I have been so busy lately I haven't been able to play but I have alreadry mapped every level so i can answer most of your questions.

First of all, you don't throw anything into the gates of hell, you break them down, enter, and get the Jeweled Fruit from the Tree of Fire. To do so you must use the Bell, Book, and Candle to break down the Gates, the Boots of Flying to be able to get back out. I have been able to get all the way out since a blast of fire gets me at the end. I suspect I need to get the Blood Blue Special (you know what I mean) to do that.

Second, you need to put the Bloodstone and the Dragon thing after evoke into the Altar, the third item I haven't found but I suspect it is the Drmpainters soul that you get after you put the third item in so it is probably the Jeweled Fruit that you need (see above).

As for Trebor, you don't find him, he finds you . Instant death.

If anyone finds the six item needed for the blue blood special tell me please (I may start p[laying again)

As for Dondra, to get out of the first room you. Get key. Open south door. Kick key south. Go South, Get key, Go North, Go North, Say Death to Colnar, Insert Key in Keyhole, North, Say Death to Colnar, go north, read mural, go south, go west. You are out in the world now. Have fun.

Data Man

<Msg #98 of 99>: Wiz IV Oracle readings

<From>: Data Man
<Date>: SUN JAN 3 7:43:21 PM

The Egress will set you free! (I believe the egress to be on Lvl 1 16N,15E)
Read the Iliad Lately?
Chomp, Chomp... Eh, what's east, Doc?
Secrets abound all around you, Have you met Glum yet? (Gives you Black Box)
Live the QABALAH!
The answer is carved in stone> It is right before you nose? (any ideas?)
The temple holds an anceint secret
Hop high to enter
Rabbits are sacred to the dreampainter
Seek the dreampainters soul
Everyone has a weekness! What is his??? (Any ideas)
Take a step to the left and a hop to the right. (refers the the stuff outside the door at the top of the temple of the dreampainter)
Gone Trolling!
Beware the gifts of Lord Maya! (I haven't met a lord Maya but he probably is the guy who gives you the "Use ME! Cape")

Get a handle on the forgidden fruit.
Rocks, multi-layered Rocks
Homer will show you the way. (Do you get the idea that we need to reread the Illiad?)
You too can be saved! Repent ye sinner! Wash away thy sins! Repent! (no idea)
Down into the bowels of the earth.
Password is your ancient Battlecry (no idea)

Ok, I have been on every level and that is all the readings I have heard (most are repeated), if you have heard any others post them. Also, if you know anything about any I said I needed info on post about that. Thanks.

PS Don't equip the following, they are cursed.
Ring of Death
Use me Cape
Adept Baldness
Mage Masher
Lord's Garb
Liches Robes
Skull;s cap

PSS Cleaning Oil removes curses, one prob, that is on level 2!

Enjoy! Data Man

==
DOCUMENT xmodem
==

MODEM PROTOCOL DOCUMENTATION

By Ward Christensen   1/1/82

I will maintain a master copy of this. Please pass on changes or suggestions via CBBS/Chicago at (312) 545-8086, CBBS/CPMUG (312) 849-1132 or by voice at (312) 849-6279.

```
Last Revision: 6/18/85  By Henry C. Schmitt.
                        State Table Appendix.

Previous Revisions: 1/13/85 By John Byrns.
                            CRC Option Addendum.

                    8/9/82  By Ward Christensen.
                            Change ACK to 06H (from 05H).
```

This version of the document was downloaded from the CBBS/CPMUG on 6/13/85 and the addition of minor editorial changes were made by Henry C. Schmitt.

Many people ask me for documentation on my modem protocol, i.e. the one used in the various modem programs in CPMUG, on volumes 6, 25, 40, 47... so here it is. At the request of Rick Mallinak on behalf of the guys at Standard Oil with IBM P.C.s, as well as several previous requests, I finally decided to put my modem protocol into writing. It had been previously formally published only in the AMRAD newsletter.

Table of Contents

1. DEFINITIONS.

<soh>	01H
<eot>	04H
<ack>	06H
<nak>	15H
<can>	18H
<C>	43H

2. TRANSMISSION MEDIUM LEVEL PROTOCOL

Asynchronous, 8 data bits, no parity, one stop bit.

The protocol imposes no restrictions on the contents of the data being transmitted. No control characters are looked for in the 128-byte data messages. Absolutely any kind of data may be sent - binary, ASCII, etc. The protocol has not formally been adopted to a 7-bit environment for the transmission of ASCII-only (or unpacked-hex) data , although it could be simply by having both ends agree to AND the protocol-dependent data with 7F hex before validating it. I specifically am referring to the checksum, and the block numbers and their ones-complement.

Those wishing to maintain compatibility of the CP/M file structure, i.e. to allow modemming ASCII files to or from CP/M systems should follow this data format:

* ASCII tabs used (09H); tabs set every 8.
* Lines terminated by CR/LF (0DH 0AH)
* End-of-file indicated by ^Z, 1AH. (one or more)
* Data is variable length, i.e. should be considered a continuous stream of data bytes, broken into 128-byte chunks purely for the purpose of transmission.
* A CP/M "peculiarity": If the data ends exactly on a 128-byte boundary, i.e. CR in 127, and LF in 128, a subsequent sector containing the ^Z EOF character(s) is optional, but is preferred. Some utilities or user programs still do not handle EOF without ^Zs.
* The last block sent is no different from others, i.e. there is no "short block".

3. MESSAGE BLOCK LEVEL PROTOCOL

Each block of the transfer looks like:

```
<SOH><blk #><255-blk #><--128 data bytes--><cksum>
```

in which:

```
<SOH>       = 01 hex
<blk #>     = binary number, starts at 01 increments by 1,
and wraps 0FFH to 00H (not to 01)
<255-blk #> = blk # after going thru 8080 "CMA" instr, i.e.
each bit complemented in the 8-bit block
number.  Formally, this is the "ones
complement".
<cksum>     = the sum of the data bytes only.  Toss any
carry.
```

4. FILE LEVEL PROTOCOL

4A. COMMON TO BOTH SENDER AND RECEIVER:

All errors are retried 10 times. For versions running with an operator (i.e. NOT with XMODEM), a message is typed after 10 errors asking the operator whether to "retry or quit".

Some versions of the protocol use <can>, ASCII ^X, to cancel transmission. This was never adopted as a standard, as having a single "abort" character makes the transmission susceptible to false termination due to an <ack> <nak> or <soh> being corrupted into a <can> and cancelling transmission.

The protocol may be considered "receiver driven", that is, the sender need not automatically re-transmit, although it does in the current implementations.

4B. RECEIVE PROGRAM CONSIDERATIONS:

The receiver has a 10-second timeout. It sends a <nak> every time it times out. The receiver's first timeout, which sends a <nak>, signals the transmitter to start. Optionally, the receiver could send a <nak> immediately, in case the sender was ready. This would save the initial 10 second timeout. However, the receiver MUST continue to timeout every 10 seconds in case the sender wasn't ready.

Once into a receiving a block, the receiver goes into a one-second timeout for each character and the checksum. If the receiver wishes to <nak> a block for any reason (invalid header, timeout receiving data), it must wait for the line to clear. See "programming tips" for ideas.

Synchronizing: If a valid block number is received, it will be:

1) the expected one, in which case everything is fine; or
2) a repeat of the previously received block. This should be considered OK, and only indicates that the receiver's <ack> got glitched, and the sender re-transmitted;
3) any other block number indicates a fatal loss of synchronization, such as the rare case of the sender getting a line-glitch that looked like an <ack>. Abort the transmission, sending a <can>

4. FILE LEVEL PROTOCOL (cont)

4C. SENDING PROGRAM CONSIDERATIONS.

While waiting for transmission to begin, the sender has only a single very long timeout, say one minute. In the current protocol, the sender has a 10 second timeout before retrying. I suggest NOT doing this, and letting the protocol be completely receiver-driven. This will be compatible with existing programs.

When the sender has no more data, it sends an <eot>, and awaits an <ack>, resending the <eot> if it doesn't get one. Again, the protocol could be receiver-driven, with the sender only having the high-level 1-minute timeout to abort.

5. DATA FLOW EXAMPLE INCLUDING ERROR RECOVERY

Here is a sample of the data flow, sendin' a 3-block message. It
includes the two most common line hits - a garbaged block, and an
<ack> reply getting garbaged. <xx> represents the checksum byte.

```
SENDER                                       RECEIVER
                                  times out after 10 seconds
                             <---            <nak>
<soh> 01 FE -data- <xx> --->
                             <---            <ack>
<soh> 02 FD -data- <xx> --->    (data gets line hit)
                             <---            <nak>
<soh> 02 FD -data- <xx> --->
                             <---            <ack>
<soh> 03 FC -data- xx   --->
        (ack gets garbaged)  <---            <ack>
<soh> 03 FC -data- xx   --->
                             <---            <ack>
<eot>                   --->
                             <---            <ack>
```

6. PROGRAMMING TIPS.

* The character-receive subroutine should be called with a
parameter specifying the number of seconds to wait. The
receiver should first call it with a time of 10, then <nak> and
try again, 10 times.

After receiving the <soh>, the receiver should call the
character receive subroutine with a 1-second timeout, for the
remainder of the message and the <cksum>. Since they are sent
as a continuous stream, timing out of this implies a serious
like glitch that caused, say, 127 characters to be seen instead
of 128.

6. PROGRAMMING TIPS (cont)

* When the receiver wishes to <nak>, it should call a "PURGE"
subroutine, to wait for the line to clear. Recall the sender
tosses any characters in its UART buffer immediately upon
completing sending a block, to ensure no glitches were
misinterpreted.

The most common technique is for "PURGE" to call the character
receive subroutine, specifying a 1-second timeout, and looping
back to PURGE until a timeout occurs. The <nak> is then sent,
ensuring the other end will see it.

* You may wish to add code recommended by John Mahr to your
character receive routine - to set an error flag if the UART
shows framing error, or overrun. This will help catch a few
more glitches - the most common of which is a hit in the high
bits of the byte in two consecutive bytes. The <cksum> comes
out OK since counting in 1-byte produces the same result of
adding 80H + 80H as with adding 00H + 00H.

7. OVERVIEW OF CRC OPTION

The CRC used in the Modem Protocol is an alternate form of block check which provides more robust error detection than the original checksum. Andrew S. Tanenbaum says in his book, Computer Networks, that the CRC-CCITT used by the Modem Protocol will detect all single and double bit errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of 18-bit and longer bursts.

The changes to the Modem Protocol to replace the checksum with the CRC are straight forward. If that were all that we did we would not be able to communicate between a program using the old checksum protocol and one using the new CRC protocol. An initial handshake was added to solve this problem. The handshake allows a receiving program with CRC capability to determine whether the sending program supports the CRC option, and to switch it to CRC mode if it does. This handshake is designed so that it will work properly with programs which implement only the original protocol. A description of this handshake is presented in section 10.

8. MESSAGE BLOCK LEVEL PROTOCOL, CRC MODE

Each block of the transfer in CRC mode looks like:

```
<SOH><blk #><255-blk #><--128 data bytes--><CRC hi><CRC lo>
```

in which:

```
<SOH>       = 01 hex
<blk #>     = binary number, starts at 01 increments by 1,
and wraps 0FFH to 00H (not to 01)
<255-blk #> = ones complement of blk #.
<CRC hi>    = byte containing the 8 hi order coefficients of
the CRC.
<CRC lo>    = byte containing the 8 lo order coefficients of
the CRC.
```

9. CRC CALCULATION

9A. FORMAL DEFINITION OF THE CRC CALCULATION

To calculate the 16 bit CRC the message bits are considered to be the coefficients of a polynomial. This message polynomial is first multiplied by X^16 and then divided by the generator polynomial (X^16 + X^12 + X^5 + 1) using modulo two arithemetic. The remainder left after the division is the desired CRC. Since a message block in the Modem Protocol is 128 bytes or 1024 bits, the message polynomial will be of order X^1023. The hi order bit of the first byte of the message block is the coefficient of X^1023 in the message polynomial. The lo order bit of the last byte of the message block is the coefficient of X^0 in the message polynomial.

9. CRC CALCULATION (cont)

9B. EXAMPLE OF CRC CALCULATION WRITTEN IN C

This function calculates the CRC used by the "Modem

Protocol". The first argument is a pointer to the message block. The second argument is the number of bytes in the message block. The message block used by the Modem Protocol contains 128 bytes.

The function return value is an integer which contains the CRC. The lo order 16 bits of this integer are the coefficients of the CRC. The lo order bit is the lo order coefficient of the CRC.

```
int calcrc(ptr, count) char *ptr; int count; {

int crc, i;

crc = 0;
while(--count >= 0) {
crc = crc ^ (int)*ptr++ << 8;
for(i = 0; i < 8; ++i)
    if(crc & 0x8000)
        crc = crc << 1 ^ 0x1021;
    else
        crc = crc << 1;
}
return (crc & 0xFFFF);
}
```

10. FILE LEVEL PROTOCOL, CHANGES FOR COMPATIBILITY

10A. COMMON TO BOTH SENDER AND RECEIVER:

The only change to the File Level Protocol for the CRC option is the initial handshake which is used to determine if both the sending and the receiving programs support the CRC mode. All Modem Programs should support the checksum mode for compatibility with older versions.

A receiving program that wishes to receive in CRC mode implements the mode setting handshake by sending a <C> in place of the initial <nak>. If the sending program supports CRC mode it will recognize the <C> and will set itself into CRC mode, and respond by sending the first block as if a <nak> had been received. If the sending program does not support CRC mode it will not respond to the <C> at all.

10. FILE LEVEL PROTOCOL, CHANGES FOR COMPATIBILITY (cont)

10A. COMMON TO BOTH SENDER AND RECEIVER (cont)

After the receiver has sent the <C> it will wait up to 3 seconds for the <soh> that starts the first block. If it receives a <soh> within 3 seconds it will assume the sender supports CRC mode and will proceed with the file exchange in CRC mode. If no <soh> is received within 3 seconds the receiver will switch to checksum mode, send a <nak>, and proceed in checksum mode.

If the receiver wishes to use checksum mode it should send an initial <nak> and the sending program should respond to the <nak> as defined in the original Modem Protocol.

After the mode has been set by the initial <C> or <nak> the protocol follows the original Modem Protocol and is identical whether the checksum or CRC is being used.

10B. RECEIVE PROGRAM CONSIDERATIONS:

There are at least 4 things that can go wrong with the mode setting handshake:

1. the initial <C> can be garbled or lost.
2. the initial <soh> can be garbled.
3. the initial <C> can be changed to a <nak>.
4. the initial <nak> from a receiver which wants to receive in checksum can be changed to a <C>.

The first problem can be solved if the receiver sends a second <C> after it times out the first time. This process can be repeated several times. It must not be repeated a too many times before sending a <nak> and switching to checksum mode or a sending program without CRC support may time out and abort.

Repeating the <C> will also fix the second problem if the sending program cooperates by responding as if a <nak> were received instead of ignoring the extra <C>.

It is possible to fix problems 3 and 4 but probably not worth the trouble since they will occur very infrequently. They could be fixed by switching modes in either the sending or the receiving program after a large number of successive <nak>s. This solution would risk other problems however.

10. FILE LEVEL PROTOCOL, CHANGES FOR COMPATIBILITY (cont)

10C. SENDING PROGRAM CONSIDERATIONS.

The sending program should start in the checksum mode. This will insure compatibility with checksum only receiving programs. Anytime a <C> is received before the first <nak> or <ack> the sending program should set itself into CRC mode and respond as if a <nak> were received.

The sender should respond to additional <C>s as if they were <nak>s until the first <ack> is received. This will assist the receiving program in determining the correct mode when the <soh> is lost or garbled. After the first <ack> is received the sending program should ignore <C>s.

11. DATA FLOW EXAMPLES WITH CRC OPTION

11A. RECEIVER HAS CRC OPTION, SENDER DOESN'T

Here is a data flow example for the case where the receiver requests transmission in the CRC mode but the sender does

not support the CRC option. This example also includes
various transmission errors. <xx> represents the checksum
byte.

```
SENDER                                      RECEIVER
                             <---            <C>
                              times out after 3 seconds
                             <---            <nak>
<soh> 01 FE -data- <xx> --->
                             <---            <ack>
<soh> 02 FD -data- <xx> --->    (data gets line hit)
                             <---            <nak>
<soh> 02 FD -data- <xx> --->
                             <---            <ack>
<soh> 03 FC -data- <xx> --->
        (ack gets garbaged)  <---            <ack>
                             times out after 10 seconds
                             <---            <nak>
<soh> 03 FC -data- <xx> --->
                             <---            <ack>
<eot>                   --->
                             <---            <ack>
```

11. DATA FLOW EXAMPLES WITH CRC OPTION (cont)

11B. RECEIVER AND SENDER BOTH HAVE CRC OPTION

Here is a data flow example for the case where the receiver
requests transmission in the CRC mode and the sender
supports the CRC option. This example also includes
various transmission errors. <xxxx> represents the 2 CRC
bytes.

```
SENDER                                        RECEIVER
                               <---            <C>
<soh> 01 FE -data- <xxxx> --->
                               <---            <ack>
<soh> 02 FD -data- <xxxx> --->    (data gets line hit)
                               <---            <nak>
<soh> 02 FD -data- <xxxx> --->
                               <---            <ack>
<soh> 03 FC -data- <xxxx> --->
        (ack gets garbaged)    <---            <ack>
                             times out after 10 seconds
                               <---            <nak>
<soh> 03 FC -data- <xxxx> --->
                               <---            <ack>
<eot>                     --->
                               <---            <ack>
```

Apendix 1. MODEM PROTOCOL STATE TABLE

A1A. CONSIDERATIONS

The Modem Protocol can be considered a group of states and
transitions. States represent certain actions taken by the
program and certain expected results for those actions.
The transitions are actions taken in reponse to a

particular result, actions which can result in another state.

The state table shows the complete set of states for a program with the CRC option. Programs without this option should ignore the <C> result in the Send-Init state and also ignore the Rec-Init-CRC state.

Apendix 1. MODEM PROTOCOL STATE TABLE (cont)

A1A. CONSIDERATIONS (cont)

There is a minor difference between the Data Flow Examples given by Ward Christensen and John Byrns. This difference is the reaction of the sender when the <ACK> to a block is garbled (not lost). In Ward's example the sender reacts by retransmitting the current block. In John's example the garbled <ACK> is ignored and nothing happens until the reciever has a timeout and sends a <NAK>. The state table uses the first method of reacting to a garbled <NAK>. This is the recommended method as the retransmission of a data block, even at the lowest baud rates, takes considerably less time than waiting for a timeout from the receiver.

In the State Table, n is the current block number (therefore n-1 is, of course, the previous block number); r is the retry counter and c is the CRC handshake retry counter. The actions n+, r+ and c+ are incrementing the appropriate counter. It should be noted that the action n+ will always cause r = 0 or, to put it another way, whenever a block is successfully sent and recieved the retry counter is reset. When a r+ action causes r to reach the threshold, an error is generated and the program is aborted.

A Result in angle brackets (i.e. < >) is the reciept of that character. A Result of "Block..." is the reciept of a complete, valid data block. Results of Other and Timeout are the reciept of any unlisted input (invalid or incomplete blocks included) and the occurance of a timeout in the character recieve routine, respectively.

This is because some installations (e.g. CompuServe) will send an <EOT> to signal that the processor is too busy to successfully transfer a file.

Apendix 1. MODEM PROTOCOL STATE TABLE (cont)

A1B. STATE TABLE

```
State
   Action on entry
      Result      Action on result                   Next State
Send-Init
   Set checksum mode, n = 0
      <NAK>       Get data for first block, n+       Send-Data
      <C>         Set CRC mode, get data
                  for first block, n+                Send-Data
      Other       r+                                    Send-Init
      Timeout     Error                                 Abort
Send-Data
   Send Block n
      <ACK>       Get data for next block, n+           Send-Data, or
                                                        Send-EOT, if EOF
      <NAK> or
      Other       r+                                    Send-Data
      Timeout     Error                                 Abort
Send-EOT
   Send <EOT>
      <ACK>       --                                    Exit
      Other       r+                                    Send-EOT
      Timeout     Error                                 Abort
Rec-Init-CRC
   Set CRC mode, Send <C>, n = 1
      Block n     Store data, send <ACK>, n+            Rec-Data
      <EOT>       Error                                 Abort
      Other       r+                                    Rec-Init-CRC
      Timeout     c+                                    Rec-Init-CRC
      c+ threshold                                      Set checksum
                                                        mode,
                                                        r = 0
                                                        Rec-Init-Cksm
Rec-Init-Cksm
   Send <NAK>
      All         --                                    Rec-Data
Rec-Data
   --
      Block n     Store data, send <ACK>, n+            Rec-Data
      Block n-1   Send <ACK>, r+                        Rec-Data
      <EOT>       If n = 1, Error                       Abort
                  Else  Send <ACK>                      Exit
      Other or
      Timeout    Send <NAK>, r+                         Rec-Data
Abort
   Display error, clean up, abort program
Exit
   Clean up, exit program
```

```
==============================================================================
DOCUMENT ymodem.s
==============================================================================

* lst off
*---------------------------------------------------------*
* Ymodem Driver source code for GBBS... orginally by     *
* Mike Golazewski or Greg Schaefer (you choose). Sourced *
* (Disassembled) w/ Merlin Pro.                          *
* This file is NOT for public distribution.              *
* (So Lance doesn't get pissed...Aw poor baby)           *
*---------------------------------------------------------*

 ORG $5000
 ORG $9E00

CHRGET = $03B1 ;acos get character routine
GETBYT = $0380 ;get next byte from segment
CHKBYT = $0383 ;check next byte in segment
GOBCOM = $0386 ;gobble a comma in segment
MOVNAME = $038F ;move filename to acos internal
SETOVEC = $03A1 ;set acos output vector to dev #
DECOUT = $03A7 ;print a signed decimal # to dev
OPEN = $03AD ;open a file using acos
CLOSE = $03B0 ;close a file using acos
RDBLK = $03B9 ;acos read a block call
ACOPTHLO = $03CB ;acos path pointer high part
ACOPTHHI = $03CC ;acos path pointer lo part
ACOSREF = $03CD ;acos reference number
PRINT = $0906 ;print to sysops screen (local)
MDMIN = $0E15 ;modem: receive a character
MDMOUT = $0E18 ;modem: send a character
MDMDCD = $0E1B ;modem: check for carrier loss
H9E00 = $9E00
H9E01 = $9E01
H9E02 = $9E02
HA5FF = $A5FF
HA600 = $A600
HA640 = $A640
HA642 = $A642
MLI = $BF00 ;prodos MLI dispatch point
KEY = $C000
STROBE = $C010
PTRIG = $C070

*----------------------------------------------------------*

 JSR CHKBYT
 CMP #$AC ;Is it a comma?
 BEQ H4F11 ;yes, go send a file
 JSR GETBYT ;no, skip past the offending character
 JSR H5174 ;zero out some counters and stuff
 JMP H4F70 ;abort the xfer
 RTS

H4F11 JSR GOBCOM ;gobble the comma
 JSR MOVNAME ;move the name to acos's buff's
 JSR OPEN ;open it using acos
 BCC H4F1D ;if all is ok, continue
 RTS  ;maybe file missing, return...

*-----------------------------------------------------------*
* Go get some file information for use in the header pack *
*-----------------------------------------------------------*

H4F1D LDA ACOPTHLO
 STA H522E
 LDA ACOPTHHI
 STA H522F
 JSR MLI ;dispatch the call
 HEX C4 ;Get File Info call
 DA H522D ;parameters found here
 BCC H4F34 ;if everything's ok, continue
 JMP CLOSE ;otherwise close it, leave

H4F34 LDA ACOSREF ;get acos internal file ref #
 STA H5240 ;store it for parms
 JSR MLI
 HEX D1 ;Get EOF call
 DA H523F ;address of Get EOF parms

 JSR H5174 ;zero out some locations
 LDA ACOPTHLO ;get pointer to filename/lo
 STA $00 ;set up indirect address
 LDA ACOPTHHI ;get pointer to filename/hi
 STA $01 ;finish setting up
 LDY #$00
 LDA ($00),Y ;get filename length byte
 TAX
H4F52 INY
 LDA ($00),Y ;get character in filename
 STA H4F8C,Y ;store it in the [     ]
 STA HA5FF,Y ;store it in the header packet
 DEX
 BNE H4F52
 LDA #$1D ;what's this?
 STA HA640

 LDX #$00 ;move the GFI and Get EOF
H4F65 LDA H522D,X ;results to header packet
 STA HA642,X
 INX
 CPX #$17 ;done all?
 BNE H4F65 ;no, loop
H4F70 LDA #$00
 STA STROBE ;clear the keyboard strobe
 STA $24 ;start flush left
 LDY #$03 ;get output channel
 JSR SETOVEC ;channel 3, sysop (local)

 JSR SPRINT ;print this, return after brk
H4F7E ASC '['
H4F8C ASC '               ] _ #'
 BRK
 LDA BADSEND ;last file check
 BEQ H4FAA ;if ok, continue
 JMP H5086 ;close 'em, stop sending
H4FAA JSR NAKIN ;wait for a <NAK> or 'C' (wrong!)

* In theory, we should wait for just 'C' or 'CK' for ymodem
* But greg decides to do it his own way.

 BCC SENDHEAD ;if ok, continue..
 JMP H5086 ;else, close and stop sending

*-------------------------------------------------------*
* Send the Header Packet with some file information in it *
*-------------------------------------------------------*

SENDHEAD LDA #$48 ;print an 'H' for header
 JSR PRSCRN

 JSR PRCOUNT ;print # of blocks sent

 LDA #$01 ;get SOH
 JSR MDMOUT ;send it out the port
 LDA BLOCKNUM ;get protocol block #
 JSR MDMOUT ;send it out the port
 EOR #$FF ;get complement of block #
 JSR MDMOUT ;send it out the port

 LDX #$00 ;start at $a600
H4FCC LDA HA600,X ;get the header packet
 JSR MDMOUT ;send the byte
 JSR DOCRC ;compute cumulative CRC for it
 INX  ;next byte
 CPX #$80 ;done 128 bytes?
 BNE H4FCC ;no, send another

 LDA CRCHI ;get CRC Hi part
 JSR MDMOUT ;send it out the port
 LDA CRCLO ;get CRC Lo part
 JSR MDMOUT ;out the port we go

 JSR ACKIN ;Check for an ACK received
 BCS SENDHEAD ;ACK not received, resend header

 LDA HA600 ;block length of 0? (EOT)
 BNE H4FF3 ;no, send file
 JMP H5089 ;else, yes, close up and return

H4FF3 JSR ACKIN ;wait for another ACK?

H4FF6 LDA #$0A ;initialize to 10 retries
 STA RETRIES ;store counter
 INC BLOCKNUM ;next xmodem block in series

 LDA BLKLO ;get blocks sent lo
 CLC
 ADC #$08 ;blocks sent=blocks sent+8
 STA BLKLO ;store result
 LDA BLKHI ;continue to make sure we
 ADC #$00 ;included the carry bit
 STA BLKHI ;store it also

 JSR PRCOUNT ;print # of blocks sent (again)

 LDX #$00 ;lo address of read call
 LDA #$A6 ;hi address of read call
 LDY #$08 ;number of 128 byte packets
 JSR RDBLK ;read 'em
 BCS H506D ;if error, end of file, close...

*----------------------------------------------------------*
* Send a Huge, 1024 Byte packet to the othe end with CRC  *
*----------------------------------------------------------*

SEND1024 LDA #$53 ;get an 'S'
 JSR PRSCRN ;print it for the sysop

 LDA #$00 ;set up indirect address to
 STA $00 ;point to $a600
 LDA #$A6
 STA $01

 LDA #$02 ;get an STX
 JSR MDMOUT ;send it out the port
 LDA BLOCKNUM ;get current block #
 JSR MDMOUT ;send it
 EOR #$FF ;255-block #
 JSR MDMOUT ;send it

 LDX #$04 ;send 4 packs of 256 bytes
 LDY #$00
 STY CRCLO ;initialize CRC lo
 STY CRCHI ;initialize CRC hi
H5044 LDA ($00),Y ;get the byte
 JSR MDMOUT ;send it
 JSR DOCRC ;compute the cumulative CRC
 INY  ;next byte
 BNE H5044 ;done 256? no, do some more

 INC $01 ;yes, next 256 bytes
 DEX  ;are we done with the 4 packs?
 BNE H5044 ;no, go send some more

 LDA CRCHI ;get CRC hi
 JSR MDMOUT ;send it
 LDA CRCLO ;get CRC lo
 JSR MDMOUT ;send it

 JSR ACKIN ;check for an ACK
 BCC H4FF6 ;ok, send the next 1024 byte pack

 DEC RETRIES ;count number of times packet sent
 BNE SEND1024 ;if count <> 10, try again
 JMP H5086 ;aborted transfer

*----------------------------------------------------------*
* End of transmission of one file, return to caller...    *
```

```
*----------------------------------------------------------*

H506D LDA #$0A ;initialize count for last byte
 STA RETRIES
H5072 LDA #$46 ;get an 'F' (final)
 JSR PRSCRN ;print it to sysop

 LDA #$04 ;get an EOT
 JSR MDMOUT ;send it
 JSR ACKIN ;wait for an ACK
 BCC H5086 ;if ok, finish up
 DEC RETRIES ;no ACK, try again...
 BNE H5072 ;if retries <> 10, try it again

H5086 JSR CLOSE ;otherwise close it, finish up

H5089 LDX #$00 ;erase bottom line
 STX $24 ;horizontal position = flush left
 LDA #$20 ;print a whole line of spaces
H508F JSR PRINT
 INX
 CPX #$27 ;done yet?
 BCC H508F ;nope, more spaces

 LDX #$0F ;put something consistent
H5099 STA H4F8C,X ;over top of the filename
 DEX
 BNE H5099
 LDA #$00 ;start flush left on return
 STA $24
 LDY #$00 ;output device is #0
 JSR SETOVEC
 RTS  ;return to calling program

*----------------------------------------------------------*
* NAKIN routine gets a <NAK>, or times out waiting        *
*----------------------------------------------------------*

NAKIN LDA #$57 ;put a 'W' on sysop's screen
 JSR PRSCRN

 LDY #$3C
H50B0 JSR INPUT ;get a character from the modem
 CMP #$15 ;is it a <NAK>?
 BEQ H50CF ;yes, return gracefully
 CMP #$43 ;is it a 'C'?
 BEQ H50CF ;yes, also return gracefully
 CMP #$03 ;is it a ????
 BEQ H50C6 ;non-fatal error in transmission
 CMP #$18 ;<CAN> character, major problems
 BEQ H50C8 ;uh oh, major type problems
 DEY  ;keep trying
 BNE H50B0 ;not done yet, try again
H50C6 SEC  ;either timed out or non-fatal
 RTS

H50C8 LDA #$FF ;fatal transmission error
 STA BADSEND ;cancel next file transmission
 SEC
 RTS

H50CF CLC  ;<NAK> received, return properly
 RTS

*----------------------------------------------------------*
* ACKIN routine gets an <ACK> , or times out waiting       *
*----------------------------------------------------------*

ACKIN LDA #$57 ;print a 'W' to sysop
 JSR PRSCRN

 LDY #$0A ;10 total for retries
 STY LASTCHAR
H50DB JSR INPUT ;get a character from the modem
 CMP #$15 ;is it a <NAK>?
 BEQ H50ED ;eww, yes, probably bad block
 CMP #$43 ;is it a 'C'?
 BEQ H50ED ;yes, bad block, or sync error
 CMP #$06 ;is it an <ACK>?
 BEQ H50F4 ;yes, return ok
 DEY  ;none of the above,
 BNE H50DB ;try again
H50ED LDA #$45 ;put an 'E' on the sysop's end
 JSR PRSCRN
 SEC  ;flag for bad data
 RTS

H50F4 CLC  ;data ok, return
 RTS

*----------------------------------------------------------*
* Get Input from modem or keyboard... <ESC> aborts send    *
*----------------------------------------------------------*

INPUT LDA #$00 ;initialize outer loop
 STA LOOPSML
 LDA #$64 ;initialize inner loop
 STA LOOPLRG
H5100 BIT PTRIG
 JSR MDMDCD ;are we still connected?
 BCC H513A ;no, close everything and return
 JSR MDMIN ;yes, get a character
 BCC H511E ;no character to get, branch
 CMP #$03 ;is it an ETX (End of Text)
 BNE H5115 ;no, check last character
 CMP #$18 ;is it a can character
 BNE H511A ;no, make this the last character
H5115 CMP LASTCHAR ;is this the last character? (can)
 BEQ H513A ;yes, flag send as bad, end it
H511A STA LASTCHAR ;no, make this the last character
 RTS

H511E LDA KEY ;check the keyboard
 BMI H512A ;key pressed? No, next part of loop
 STA STROBE ;yes, clear strobe
 CMP #$1B ;is it an <ESC>?
 BEQ H513A ;yes, stop transfer
H512A BIT PTRIG ;???
 DEC LOOPSML ;take car of inside loop
 BNE H5100
 DEC LOOPLRG ;take care of large loop
 BNE H5100 ;not done, try some more
 LDA #$00 ;done, nothing, 1 timeout
 RTS

*----------------------------------------------------------*
* Take care of bad send info... abort transfer             *
*----------------------------------------------------------*

H513A CMP #$18 ;is it a can character?
 BNE H5143 ;no, just go abort it
 LDA #$FF ;<CAN>, so mark it as bad send
 STA BADSEND ;save it
H5143 PLA
 PLA
 PLA
 PLA
 JMP H5086

*----------------------------------------------------------*
* Calculate a cumulative CRC-16 from Accumulator           *
*----------------------------------------------------------*

DOCRC PHA
 EOR CRCHI
 STA CRCHI
 TXA
 PHA
 LDX #$08
H5155 ASL CRCLO
 ROL CRCHI
 BCC H516D
 LDA CRCHI
 EOR #$10
 STA CRCHI
 LDA CRCLO
 EOR #$21
 STA CRCLO
H516D DEX
 BNE H5155
 PLA
 TAX
 PLA
 RTS

*----------------------------------------------------------*
* Initialize some locations & counters                     *
*----------------------------------------------------------*

H5174 LDA #$00 ;zero out some counters
 STA BLOCKNUM
 STA BLKLO
 STA BLKHI
 STA CRCLO
 STA CRCHI
 TAX
H5186 STA HA600,X ;zero out 1024 bytes
 INX
 CPX #$80
 BNE H5186
 RTS

PRSCRN PHA  ;save accumulator
 LDA #$12 ;save as horizontal position
 STA $24
 PLA  ;restore acc
 JSR PRINT ;print character in acc at horiz
 RTS

SETUP LDA H9E00
 CMP #$4C
 BEQ H51CC
 LDA #$4C
 STA H9E00
 LDA #$00
 STA H9E01
 LDA #$4F
 STA H9E02
 LDA $04
 PHA
 LDA $05
 PHA
 LDA #$2C
 STA $04
 LDA #$52
 STA $05
 JSR GETBYT
 PLA
 STA $05
 PLA
 STA $04
 LDA #$00 ;flag it as no bad send
 STA BADSEND
 RTS

H51CC LDA #$00
 STA H9E00
 STA H9E02
 LDA $04
 PHA
 LDA $05
 PHA
 LDA #>H522C
 STA $04
 LDA #<H522C
 STA $05
 JSR GETBYT
 PLA
 STA $05
 PLA
 STA $04
 RTS
```

```
*----------------------------------------------------------*
* Print number of blocks to sysop's screen (local)         *
*----------------------------------------------------------*

PRCOUNT PHA  ;save the ACC
 TXA  ;save the x reg
 PHA
 LDA #$15
 STA $24 ;new horizontal position
 LDX BLKLO ;block count lo
 LDA BLKHI ;block count hi
 JSR DECOUT ;print it
 PLA  ;get X
 TAX
 PLA  ;get ACC
 RTS

*----------------------------------------------------------*
* Here begins a fairly sophisticated print routine, more   *
* sophisticated than necessary in my opinion..             *
*----------------------------------------------------------*

SPRINT PLA  ;get return addr hi
 STA $48
 PLA  ;get return addr lo
 STA $49
 TYA
 PHA  ;save the y reg
 LDY #$00
 BEQ H520F ;does this work?
H520C JSR PRINT
H520F INC $48
 BNE H5215
 INC $49
H5215 LDA ($48),Y
 BNE H520C
 PLA  ;restore Y
 TAY
 LDA $49 ;get changed return addr lo
 PHA  ;push it
 LDA $48 ;get changed return addr hi
 PHA  ;push it
 RTS  ;return to altered location

*----------------------------------------------------------*
* Temporary Storage locations used by the program...       *
*----------------------------------------------------------*

BLKLO HEX 00 ;blocks sent lo
BLKHI HEX 00 ;blocks sent hi
CRCLO HEX 00 ;CRC lo
CRCHI HEX 00 ;CRC hi
LOOPSML HEX 00 ;small loop
LOOPLRG HEX 00 ;large loop (for input)
LASTCHAR HEX 00 ;last character sent
RETRIES HEX 00 ;# of retries
BLOCKNUM HEX 00 ;block number
```

```
BADSEND HEX 00 ;condition of last file sent
H522C HEX 00 ;???
H522D HEX 0A
H522E BRK
H522F BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
 BRK
H523F BRK
H5240 BRK
 BRK
 BRK
 BRK
```

==
DOCUMENT zmodem.gbbs
==

Zmodem comes to GBBS!

This past summer, between writing different versions of Shrinkit and GS-ShrinkIt, I wrote Zmodem drivers for several different bulletin board systems. If you are the sysop of a bulletin board which uses GBBS's ACOS language you can take advantage of a very good deal.

For $21, I will send you a copy of the Zmodem drivers which work with GBBS.

But before I go into a little more detail about this, let me explain a little more about the Zmodem drivers.

Features:

o Both RZ and SZ completely conform to the public domain Zmodem 2.0 implementation by Chuck Forsberg using 16-bit CRCs.

Both Zmodem Send (SZ) and Zmodem Receive (RZ) are completely and correctly implemented and take approximately 4k of space in the GBBS "use" buffer along with an extra 8k of buffer space in auxiliary memory.

o There are special versions of RZ and SZ which use the Apple IIc's vertical-blanking interrupts for timing considerations.

o Speed. Zmodem is a streaming protocol. This allows for faster transfers than Ymodem and helps 9600 baud transfers go close to their theoretical maximum.

o Better error recovery. Zmodem can recover from errors better than Xmodem or Ymodem. If you have a really noisy line, chances are that Zmodem will continue the transfer long after Xmodem and Ymodem have given up.

o Network friendly. RZ and SZ will not "jam" a network by sending XOFF characters in its data stream. Instead these characters are sent using Zmodem's escaping mechanism. What this means is that you won't have to setup your local node of PcPursuit or other service when calling a BBS that uses RZ and SZ. No special parameters for your node should be needed. Just call and transfer.

o Zmodem is a "batch" protocol. Both drivers support sending and receiving batches of files.

o Auto-Download support. If your terminal program supports Auto-Download, then using RZ with your BBS will automatically tell your communications software to begin downloading without ever touching a key!

o Download resumption. If you have a communications program which supports resuming a download after you have been disconnected while downloading a huge file... no problem. Just call back and begin the download at the point where you left off. These Zmodem drivers properly support doing just that.

o SZ (the BBS end receiving a file via Zmodem) supports both upload resumption (if you, as a sysop, like incomplete pieces of files laying around on your BBS), renaming an existing file, or just deleting a file on the BBS which the user is trying to upload. So, in the case of a duplicate file, you have great flexibility in what to do.

o Automatic block resizing! Xmodem sends files with 128 byte blocks. Ymodem sends files with both 128 byte and 1k blocks. Zmodem can use any block size up to 1k. When downloading, RZ will take note of how noisy the phone line is and if there are enough errors RZ will halve the block size until some data gets through.

   If you have clean phone lines then RZ will start increasing the block size until it is streaming 1k blocks. So, the cleaner your phone lines are: the faster your transfers will be.

   If during the course of a download the phone lines become very noisy then RZ will make the blocks smaller -- and if the line becomes less noisy later during the download, RZ will start sending larger blocks.

* Just a note. At this time, although plenty of Macintosh and IBM PC communications programs like ZTerm and ZComm support Auto-Download and file resumption, I do not know of any Apple II communications software which does. ProTerm 2.2 and prior do not support Auto-Download or file resumption, although it is conceivable that ProTerm 3.0 will (we can hope).

```
 What you need to use RZ and SZ:
 ------------------------------

      An enhanced (65c02) 128k Apple IIe, IIc, or Apple IIGS
      GBBS "Pro" 1.3 or later (preferably later)
      A good working knowledge of GBBS's language, ACOS.

 What your $21 will get you:
 --------------------------

      SZ and SZC (Send Zmodem and Send Zmodem for the Apple IIc)
      RZ and RZC (Receive Zmodem and Receive Zmodem for the Apple IIc)
      Notes on how to write a simple module for your BBS to support Zmodem
       transfers.

      The latest versions of ShrinkIt, GS-ShrinkIt, II+ ShrinkIt, and
       AUTO-Unshrinkit will be included as a bonus (since they are, after all,
       freely available).
```

Because I am keenly aware of the amount of piracy that a product like this will undergo, the following stipulations have to be attached:

o Please pay by check. Orders received in cash will be returned.

o I will wait up to 3 months until I have received 40 orders before shipping anyone's order. This means that the sooner I receive 40 orders, the sooner everyone will receive their copy of Zmodem for GBBS. If you are not prepared to wait a while -- because I can't predict how long it will take to receive 40 orders -- then please do not order this.

o If I do not receive 40 orders, I will return everyone's checks uncashed.

o I am not going to attempt to hunt down those who choose to illegally distribute what I write -- I would only hope that some of them have the decency to pay for what they use. There isn't any tomfoolery in the drivers either. No secret codes or encryption or serial numbers. If you buy a copy, I will send you a copy. It's as simple as that.

If this sounds reasonable to you, then send a check for $21 to:

        Andy Nicholas
        1180 Reed Ave, Apt 12
        Sunnyvale, CA 94086

and make sure you specify what kind of disk (3.5" or 5.25") on which you need the Zmodem drivers -and- where to send the Zmodem drivers.

About the Author:

I've written the freeware programs ShrinkIt, GS-ShrinkIt, ShrinkIt for the Apple II+, and AUTO-UnShrinkIt (shrinkit archive scavenger/extractor) and am currently employed by Apple Computer to work on the Apple IIGS Finder. This is work that I did before coming to Apple and work that I'm doing in my spare time. I believe in low-cost, high-quality software. I also believe in trying to get that software to as many people as possible.

At this time (1/27/91), there are tentative plans for the distribution of Zmodem drivers for both ProLine and the Prime BBS system. These Zmodem drivers have already been written and tested, although I will almost certainly not handle their distribution.

If you have questions about the Zmodem drivers, suggestions for future versions of ShrinkIt, or suggestions for the Apple IIGS Finder, I can be contacted on America-Online, GEnie, CompuServe, and the internet at:

```
America-Online & Genie: shrinkit
            CompuServe: 70771,2615
              Internet: shrinkit@apple.com
```

F I N I S