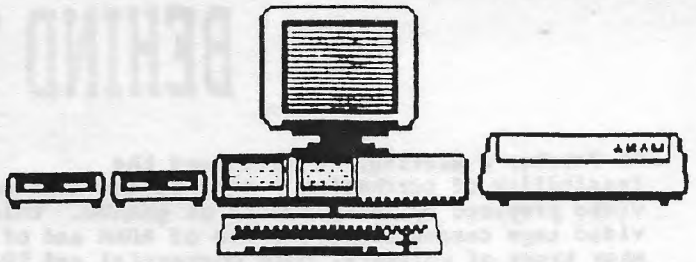


NOVEMBER 1987

EDITOR: PATRICIA HERRINGTON GRAPHICS BY: MICHAEL GRAHAM

Dear Fellow ADAMite,

Once again, there's way too much news to go in one little newsletter. I haven't been able to find space to review the news from around the country; but, as usual, I will bring the newsletters from other groups to the meeting for you to explore. So much is happening... and all our sister groups and third party developers are busy making sure the news never gets stale. Great things are in the works!

As Thanksgiving approaches, I know that my traditional moment of pausing to reflect on all I have to be thankful for will, this year, include thanks for all the new friends I have made since I have been working on this newsletter. That means I'm grateful for all the folks I've "met" around the country, and also for each and every one of YOU, my readers, who make me feel useful... that's not to be sneezed at!

And after Thanksgiving... Christmas! We have a few things up our sleeves to make your holiday season brighter. I love surprises, so I'm not going to give you hints... but I think you'll be pleased.

This month's BONUS will be a CRUNCHER program. See inside for an explanation of what that means! You may already have one, but you probably will want the updated version just the same.

Next meeting is:

NOVEMBER 8 — Orlando Public Library 1:30 PM

Third floor, in one of the conference rooms (probably the Cypress Room.) Call Allen & Frances Bell if you want to order something. And call John Terry or Mike Graham with any other questions, suggestions, or information. Or write me... I'd love to get your input... at the address on the back cover. And, please join us !!!

BELLS: 352-0724 TERRYS: 857-6635 GRAHAM: 323-2781

See you Sunday!

Pat

BEHIND THE SCENES

At October's meeting, we discussed the feasibility of purchasing a copy of the ADAM demo video prepared by Terry Fowler of gHAAUG. This video tape demonstrates the use of ADAM and of many types of software, both commercial and PD, including the word processor. We also discussed buying the gHAAUG public domain library; Terry sells this very reasonably, and it is already cataloged. We would then have access to a large number of PD volumes which we wouldn't have to worry about cataloging. Thomas Hartter suggested that; if we each pitched in $5 or $10, the low price would be within our range. This is a good idea; now that we're pursuing tax-exempt status as a non-profit group, donations would be deductible on our income taxes. Rick Covell started the fund going, and he also suggested that we make a tax-free donation of our own: we can donate the above-mentioned videotape to the library, bearing information about MOAUG. We may well attract new members to the circle.

We met two new members who have never before been to a meeting; Gordon Moseley has subscribed for some time, but never been able to join a meeting before. Ric Stilfield drove all the way in from Melbourne to see us! Ric has volunteered to bring his Orphanware 80-column card to a future meeting, if we can find someone to bring a monochrome monitor. I'd like to see this; I plan to own one someday. It would be an enormous help in CP/M. Ric has CP/M experience to share, too.

We mentioned the possibility of holding a new users' clinic, maybe some time between Christmas and New Year's, probably at my house. I could just have an open house, and help with your questions... even demonstrate some of the copyrighted software I own to those of you who may be considering future purchases. Interested?

We distributed, as a BONUS, the final five chapters of the BASIC programming course. We now have all ten chapters available as a PD pack. We also, as usual, had some freebies to give out: E&T catalogs, Capitol printer interface catalogs, sprite sheets, game scorecards, etc. I recently reduced the instructions to Pinball Construction set; even reduced, they take up six pages! We gave these out, too, and we do have more left. For next meeting, we will also have instructions for ADAMLINK II and Speedcheck. If you've been receiving newsletters for awhile, you already have these, but otherwise you will want a copy. All these items are free. If you can't attend meetings but want any of the above, I'll send them to you; but please send me postage (1 stamp for Pinball Construction Set instructions, and 1 stamp for any one to four other items should do it.) Other handouts are in the planning stage, and Terry Fowler just sent us a couple of items.

Our Nibbles & Bits PD library is growing, too. More on what's available next issue... let me know what you're interested in, OK?

MOAUG was mentioned in this month's NIAD, and in gHAAUG News. Nibbles & Bits is running some of our articles. And Ray Dougherty is mentioned in both PSAN and gHAAUG, who were really interested in that extra screen in AdamCalc that he saw on Compuserve! (Thanks to Tom Clary for this one.)

Meanwhile, back at the ranch, Bill Strasser writes that his ADAM BBS is going back up. He has gotten a second phone line specifically for the BBS. He says it will be operating 24 hours a day every day, except that it will be down on Wednesdays for maintenance. He expects to have downloads after the first of the year. He says, "All control functions are operational. That means all interested parties should be able to print the screen this time. There'll be a 40 column program in the near future that will allow compatibility with the board as well as switching back to 32 column after you've finished accessing the board." You can use ADAMLINK's default values ... 7 character bits, 1 stop bit, even parity... in other words, you don't have to change anything in ADAMLINK: PHONE: 851-8575.

Nobody has complained yet, but this newsletter is pretty heavily slanted toward beginners. It will continue to be so as long as most of our people want this type of information. Still, I try to find items that will stimulate our more advanced members, too. There are some goodies in this issue. I think beginner and hacker alike will enjoy George Havach's Notes on Basic 2.0, and will appreciate the final installment of Rich Lefko's Copycat series. Gregg Noblett has contributed heavily to this issue... he not only found the CRUNCHER article I was looking for, he even typed it in and uploaded it! And for you CPMers, we are printing Gregg's Quick Reference cards for NULU and MADAM7.

Please note that Quick Reference cards are not intended to be lessons in themselves, but are meant to be memory-joggers for the person who is already somewhat familiar with a program. And a great idea they are, too! We all make little notes to ourselves as we go along... and for those of you who break out in hives at the very thought of writing a review, this could be the perfect way to share your findings with others. Just jot them down, we'll format 'em onto a filecard! There are always others who need them.

Rich Lefko has a 64k card for sale (he's upgrading, lucky devil) for $25. He also has a hard-to-find Coleco Expansion Module #1 (accesses Atari/Sears cartridges) for only $40... still in the box! Add shipping for either item. His address: 2621 Collier Ave. San Diego, CA 92116 (Compuserve ID: 70017,3535)

Bob Crecco has a BANANA printer, including ribbons, ink AND a parallel Centronics interface, which is everything you need to get started, for $75. (1627 Dewey St, New Albany, IND, 47150.) Phone: (812) 9848-1414 CIS ID: 70017,3535

Big John of Orphanware has exactly ONE disk drive left at $175. His address is on the back page, and he accepts COD by phone: (216) 882-4720.

Thom Andree of Orphanware Service Center repairs any and all ADAM components... reasonably. Address: 5641 Leibold, Huber Heights, OH, 45424 Phone: (513) 254-2785... Many favorable reports.

Terry Fowler of gHAAUG still has plenty of spare data drives for just $9.95. See Terry's phone & address in last month's issue. MOAUG co-treasurer Kathy Nolte just ordered two, and was very pleased with both merchandise and service. You can't beat this deal.

COPY-CAT
COPY-CAT
~PART III~

BY RICK LEFKO

I'm sure that some of those who have been reading this series of articles are wondering what copy program I use. For most of my copy needs I use "Quickopy" by Glen Garbaric (GJMG Enterprises). However, before I begin discussing Quickopy, I think it's only fair to give honorable mention to "Packcopy" by Daryl Sage (available through MW Ruth $29.95 or directly from Sage Enterprises.) Packcopy was the original copy utility for the ADAM. Packcopy can perform image block copy between DDP's or disk drives. Packcopy was, and still is, a fine copy utility, but it has its limitations and nowadays you can get more powerful copy utilities at the same or lower price.

One of these is Quickopy (available from NIAD: $15.95.) Quickopy comes with an excellent 32-page manual that explains much more than you'd care to know. Quickopy will automatically utilize your 64K memory expander, if you have one, and use a 104K copy buffer. If you don't have the 64K expander, Quickopy will use a 40K buffer. What this means is a lot less wear and tear on your drives. You see, with such a large copy buffer you will be able to copy between two disk drives in one and one-half passes, or between two data drives in two and one-half passes. Packcopy, as well as Utilcopy, utilizes a 15K buffer. That's ten and a half passes between disk drives and almost seventeen passes between data drives! EZcopy uses an 8K buffer. That's twenty times between disk drives and a whopping 32 times between data drives! So I'm sure you can see why a large buffer is desirable. Quickopy is self-booting, so just load it into a drive and push the reset button. I have it on disk, but even on DDP it loads extremely fast because it's written entirely in machine language.

After booting Quickopy the first thing you'll see is the entrance screen showing you what kind of buffer it will be using. It will adjust automatically to the 64K expander. The main menu gives you a choice between "block" copy or "file" copy. Block copy will allow you to copy an entire disk or DDP to another disk or DDP, block by block, or it will allow you to copy from a specific starting block. If you choose "file" copy you will be asked your source and destination drives. Then you'll see the volume data screen which shows the name of the volume and how many directories you have. Quickopy can handle up to 16 directories. Then, after pressing return, you'll see the file data screen. This screen is jam-packed with information about your files. It will tell you the following about EACH file: start block, blocks allocated, blocks used, bytes used. It will also give you the following info, yes or no: permanently protected, read-protected, user file, deleted file, write-protected, executable, system file.

continued, next page

At this point you have the option to choose which files you'd like to copy. You can even go back if you change your mind. When you've reached the end of the directory Quickopy displays the number of blocks left and how many files you are copying. Then you can copy or re-display. If you decide to copy, the Quickopy status screen keeps you informed as to what's going on at all times. You can also copy to and from the same drive.

I find that Quickopy fills all of my copy needs excellently. "Backup + 3.0" by MMSG ($32 MW Ruth or Alpha 1) is another fine copy utility similar to Quickopy. Backup + can also format disks. (There is a version called just Backup... without the '+'... which does not format disks; $17.95 from MW Ruth.) *

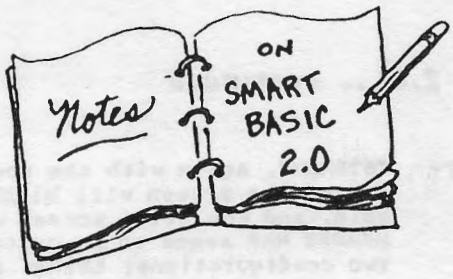
The last type of copy utility I'd like to discuss is a cartridge copy program called "Copycart +" by MMSG (available through MW Ruth or Alpha 1 $15 D/DP.) Copycart allows you to copy 9, 15 kilobyte cartridges, to disk or, 15 to data pack. When you boot this media the first menu screen will ask you if you want to make target media, or copy a cart. This may sound difficult, but believe me, it isn't. What you do is make special "target media" on another disk or DDP. Then when you copy a cart, you copy it onto the media you've created. After you've copied a cart to this medium it becomes self-booting. When you boot it a menu will display a list of the carts available. Then, all you need to do is pick the number of the cart you want and it will automatically be loaded and run. This program will copy MOST game carts. I have found a couple that wouldn't copy for me, like "Dam Busters". Copy Cart is real easy to use and will cut down on the wear and tear of your game carts.

This ends my series of articles on copy programs. I hope some of the software I have reviewed here will aid you in making an intelligent decision on which copy program to buy. Certainly, I haven't covered them all. One thing is certain though, you need to have a copy program!

Remember, back-up your software!

Rich Lefko

* Editor's note: Backup+ 3.0 by MMSG also has another feature which is especially valuable to new disk-drive owners: it will copy Basic from a data pack to disk, CHANGING IT AS IT COPIES to a disk version that will boot from the disk drive. This can be done in other ways, but Backup+ makes it easy.



Notes on SmartBASIC 2.0

by GEORGE A. HAVACH
COMPUSERVE ID# 70337,3062

Though never finally released by Coleco (who terminated the project shortly before discontinuing further Adam production and development), a revised, still incomplete (no enhanced sound or graphics!!) version of SmartBASIC dubbed 2.0 has found its way into the "Adam underground" in what amounts to a beta-testing form. Having obtained a working copy of my own, I've spent some hours checking it out and comparing its features with those of the V1.0 we've all come to know and love/hate. The following somewhat scattered notes summarize my discoveries to date.

(1) SmartBASIC II occupies 49 blocks on the disk, in contrast to SB I's mere 28K. The file size is so unexpectedly large because SB II COMES WITH ITS OWN OPERATING SYSTEM ON BOARD--true Revision 7 of EOS (different from the ROM-chip version built into the ADAM) -- which is overlaid onto the RAM-resident version at startup, beginning at address E4B8H on up. According to Joel K. Lagerquist, author of the revisions to SmartBASIC that were incorporated into V2.0, this revision comes equipped with all the fixes to EOS' file-handling functions that enable SB II to work "much faster" with data files opened by SmartBASIC programs. Thus, SB II is apparently capable of correctly opening and read/writing random-access files, whereas SB I seems full of bugs in this department, in spite of what the Coleco manual (pp. C9-C10) would lead you to believe. Furthermore, SB II takes up almost 500 bytes LESS runtime memory than SB I: a FREe space of 26401 [after NEW] versus 25954 bytes.

(2) The cold-start loader ('BOOT' block 0) is all new code and includes a RAM test for the presence of the 64K Memory Expander. Also, SB II will boot from ANY drive and set that as the active one, whereas SB I defaults to data-pack drive 1 unless patched otherwise (byte 2 of block 18: 4 = disk drive 1, 8 = data-pack drive 1).

(3) The default value of HIMEM (53632) is identical in the two versions, but LOMEM is lower in SB II: 26960 versus 27407 (the same difference as for FRE(0)). Also, the system-parameter area (see SmartBASIC Guide, p. C-23) has been moved: for example, the "LOMEM setting" is at 16095 in SB I, but at 1594 in SB II. More interesting, LOMEM and HIMEM can now be simultaneously reset; in SB I, you can change EITHER one, but then trying to change the other gives you an "Out of Memory Error."

(4) Both the REM and DATA statements have been fixed for the bug in SB I that keeps inserting a space after them on a program line each time the program is SAVEd or LOADed. In fact, SB II specifically strips out any extra leading spaces when LISTing. This single feature might make SB II worth having, just to be able to save programs within a minimum of file space.

(5) The cursor-display routine has been rewritten in SB II, so now the underline cursor character DISAPPEARS while either executing an immediate command or running a program, reappearing only on return to command mode. This makes for a "cleaner" screen display, but it's no longer possible to change or blank the cursor character with a single POKE (to 16953 in SB I).

(6) Return of any error message by SB II is accompanied by a "beep" (CHR$(7)). Furthermore, a touch of humor--and genius--has been added to the way numbered program lines are accepted (or not) by the interpreter. Previously, any line containing a syntax ("Illegal Command") error was simply rejected by SB I. Now, the offending line will still be normally redisplayed with a caret pointing to where the error occurred, but in addition, SB II prints a "happy face" (CHR$(2)) as the first character after the line number. And if you LIST the program, any erroneous lines are RETAINED IN THE PROGRAM, but their presence is unmistakably signaled by a "beep" during the listing! This means a potentially great savings of time and retyping while programming: buggy lines now can be corrected AT ANY TIME DURING PROGRAM DEVELOPMENT, simply and easily by running the cursor over the line (skipping over the "happy face" with a CONTROL-right arrow), making the needed changes, and then hitting RETURN to reenter the line and cancel the incorrect version. The program, of course, will not run properly until all these troublesome lines are fixed.

(7) Four of the special-function keys ("hardkeys") are software-labeled to function for line editing in SB II: CLEAR works like CONTROL-X (cancels a line input), INSERT like CONTROL-N (opens up a space leftward of the cursor), DELETE like CONTROL-O (closes up the space beneath the cursor), and PRINT like CONTROL-P (dumps the screen to the printer). The screen dump, however, is now INTERRUPTABLE WITH A CONTROL-C, and if the CONTROL-C is issued during a screen dump activated from within a program (e.g., by PRINT CHR$(16)), control passes to the next step without a "Break."

(8) Line length (actually, the size of the line buffer) in SB II is expanded to 255 characters, versus SB I's 127.

(9) Use is made of an RST 38 interrupt vector that involves updating the spinner (the spool on the Super Action Controller), evidently with game design in mind.

(10) The BREAK and NOBREAK commands (tokens 64 and 65) have been deleted from SB II, but STORE, RECALL, and SHLOAD have not, and they're just as NON-FUNCTIONAL as in SB I!

Continued, next page

... NOTES ON BASIC 2.0... CONTINUED

(11) The new MERGE command works beautifully, in precisely the same way as in Microsoft BASIC: "Format: MERGE <filename>[,Dn]. Merges a specified disk file into the program currently in memory....If any lines in the disk file have the same line numbers as lines in the program in memory, the lines from the file on disk will replace the corresponding lines in memory. (MERGEing may be thought of as 'inserting' the program lines on disk into the program in memory.) This single addition greatly facilitates "modular" programming, and makes SB II particularly valuable for larger writing projects.

(12) The INIT command in SB II assigns the correct number of "Blocks Free" when used to blank out the directory and reinitialize a disk, whereas SB I treats a disk the same as a data pack, and so 'CATALOG' shows an impossible "253 Blocks Free"!

(13) While SB I sets an upper limit to use of the POKE command of 54160 (D390H, the start of the FCB-header area), SB II raises this limit nearly to the top of RAM, to 65216 (FEC0H, the beginning of the DCB/FCB area). You can override this limit by POKEing 255 to addresses 16149 and 16150 in SB I, or to 1648 and 1649 in SB II; then you can POKE anywhere in memory--at your own risk!

(14) In general, special POKEs into the BASIC interpreter of SB I--and programs that use them--will NOT work in SB II, since it has been completely recompiled. In many cases, however, corresponding locations can be found with a little investigation. Among the more useful are the POKEs to change the screen and character colors (SB I addresses in parentheses):

17184 (17059) border color (0-15)
17240 (17115) normal-character (high nibble) and display-screen (low nibble) colors
17251 (17126) inverse-character (high nibble) and block-background (low nibble) colors

(15) The two new commands STDMEM and EXTMEM have been added to SB II to permit switching between standard memory (64K RAM plus 16K VRAM; also the default configuration) and extended memory, which uses the 64K Memory Expander (if present) in much the same manner as SmartWriter uses this extra RAM to expand its workspace, i.e., through a bank-switching arrangement. To load the bank-switched configuration (which takes up much of that extra space in the SB II disk file), enter "EXTMEM" WITH THE SOURCE DISK STILL IN THE DRIVE. The disk will spin, the screen will blank, and after a few seconds the title screen will reappear. To get some idea of the size of your new workspace, enter "PRINT FRE(0)"; the value 90646 is displayed [90656 after "NEW"]. Compare this with the value of 26391 [26401 after "NEW"] in SB II's standard memory configuration. To reload the standard configuration (and clear the workspace), enter "STDMEM", again with the source disk in the drive; the screen will blank, the disk will spin, and the title screen will reappear. The MEMORY MAP seems to be quite different in these two configurations; things really get moved around, so watch out! For example, LOMEM and HIMEM don't seem to have the same error checking in EXTMEM, and so entering an illegal value can cause an immediate system crash! Also, programs will generally RUN MORE SLOWLY (about half as fast) in EXTMEM than in STDMEM, because of all the extra CPU time spent in bank switching. That's the way it goes!

(16) High-resolution graphics have supposedly been fixed in SB II to eliminate the problem of "video bleed" when HPLOTing or DRAWing in adjacent pixels, but the few simple tests I've run indicate that the color-bleed problem is AS BAD AS, IF NOT WORSE THAN, IN SB I. Nonetheless, THE BIG NEWS IS THAT SB II IS CAPABLE OF HANDLING SPRITES! It does this through the "DRAW" and "XDRAW" commands--and these work IN ALL THREE SCREEN-DISPLAY MODES: TEXT, GR, and HGR/HGR2. To activate this function, first POKE a nonzero value into address 16788 (the shape/sprite flag; default value is 0); addresses 16786 and 16787 hold the pointer to the starting address of the sprite table, low byte first (the default table is at 192 decimal). A sprite table can be located anywhere in memory you choose, with its address POKEd into these two bytes. There are TWO default sprites already provided; you can view them by the following interaction:

```
POKE 16788, 1: HOME
HCOLOR = 3: DRAW 1 AT 100, 50: DRAW 2 AT
100,100
```

This will display sprite 1 as a sailboat and sprite 2 as a communications satellite! The format for creating your own sprites is EXACTLY THE SAME AS FOR SHAPES IN SMARTLOGO, as described in the SmartLogo Reference Manual (pp. 72-74); each sprite is a 32-byte string of hexcodes. A maximum of 32 sprites is allowable (the limit for simultaneous display by the video chip), numbered 1 through 32.

(17) SB II has a built-in function for displaying ANY character on the screen at the current cursor position (in TEXT mode), even the control- character symbols below ASCII 32. To utilize this function, POKE the ASCII code into address 16771, then CALL 16770. This function interprets the code as a displayable character, NOT a control code, so you can print even the characters equivalent to a linefeed (10, a Greek "pi") or carriage return (13, a left-pointing arrowhead). Try this short program:

```
10 HOME: FOR char = 1 to 127: POKE 16771,
   char: CALL 16770: PRINT " "; : NEXT
```

which will show a staggered screen display of the entire Adam character set, from ASCII 1 (a centered dot) through 127 (a flashing cursor).

.... George A. Havach

NULU CP/M

QUICK REFERENCE CARDS BY GREGG NOBLETT
FILE CARD GRAPHIC BY MIKE GRAHAM

NULU Command Menu:

A- add members
B- brief toggle
C- close the library
D- delete members
E- extract members
F- filesweep mode
G- get filespec
K- krunch library
L- list members
M- menu
N- rename members
O- open a library
P- print members
Q- unsqueeze members
R- replace members
T- replace/add members
U- drive/user change
V- view members
X- exit NULU
Y- disk directory
Z- zap disk files
> - redirect output
< - redirect input

NSWP commands:

? - menu
X - exit
:C - copy
D - delete
P - print (^C interrupts)
R - rename (* → puts into batch)
V - view
T - tag
U - untag
M - mass transfer
Q - squeeze/unsqueeze tagged
OPTIONS:
S - SQUEEZE
U - UNSQUEEZE
R - OPPOSITE OF WHAT FILE IS
A - tag again
L - log new drive
B - move back through directory
S - calculate space remaining on drive specified
W - wildcard tag of files
C (opt) - copy with option — when prompted for drive, add a 'V'.
EXAMPLE: 'B3:V'

MADAM7 CP/M

T = TERMINAL MODE (FULL DUPLEX)
L = LOCAL ECHO (HALF DUPLEX)
E = ECHO TO REMOTE (LIKE A BBS)
TERMINAL OPTIONS: O: ORIGINATE A: ANSWER

Cntrl-E: ESCAPE TO COMMAND MODE
Cntrl-Y: START COPY INTO TEXT BUFFER
Cntrl-R: STOP COPY INTO TEXT BUFFER
Cntrl-P: TOGGLE PRINTER ON/OFF
? = DISPLAY CURRENT SETTINGS
^# = FUNCTION KEY INTERCEPT. # = 0-9
M = DISPLAY MENU
R_FILENAME.EXT = START DOWNLOAD (XMODEM)
S_FILENAME.EXT: SEND FILE (UPLOAD) (XMODEM)
R (OPT)_FILENAME.EXT = DOWNLOAD W/OPTION
OPTIONS: B = BULK USING WILDCARDS (*.*)
D: DISCONNECT WHEN DONE
Q: NO STATUS REPORTS (QUIET)
V = VIEW BYTES ON CONSOLE
X: WHEN DONE, DISCONNECT → CPM

"_" MEANS "SPACE"

CPM = EXIT TO CP/M
DIR = DIRECTORY + FREE K's
ERA ERASE FILE
LOG = CHANGE DEFAULT & RESET DISKS
TCC TOGGLE CRC/CHECKSUM
TLF = TOGGLE LINEFEED AFTER <CR> IN L OR T MODE
NUM = LIST REMOTE SYSTEMS
BYE DISCONNECT, THEN → CP/M
CAL CALL PHONE #
DSC DISCONNECT FROM PHONE
DEL = DELETE TEXT BUFFER
WRT = WRITE TEXT BUFFER TO DISK
Cntrl-X = ABORT CALL FUNCTION BEFORE CONNECTED
TXO = TOGGLE XON/XOFF TESTING
SPD SET FILE OUT SPEED IN TERMINAL MODE
TIM = BAUD RATE FOR "TIME TO SEND" MSG
TLC TOGGLE LOCAL COMMAND (IMMEDIATE / AFTER CONTROL)

CREATE TEXT BUFFER:

^E,T_ FILENAME.EXT, EOB

ASL USES: CHECKSUM

COMPUSERVE USES: CHECKSUM/CRC

+ FUNCTION KEYS:

1 - GO EASY
2 - GO CLUB
3 - GO CPMSIG
4 - 70317,3207
5 - (CI$ PASSWORD)
6 - (ASL PASSWORD)
7 - GREGG NOBLETT
8 - GO CHARGES
9 - ** I HOPE THIS HELPS **
0 - /EXIT

SPEED LOADING

One problem common to all ADAM owners (at least until we break down & get a disk drive!) is that Basic programs are so S-L-O-W to load. Actually, this is nothing more than an inconvenience, and we learn to cope. We know that while our favorite file is loading, we can go get a cup of coffee. And if it's a nice, hefty text game like MOTHERLODE, we can go to the store, buy the coffee, and make a fresh pot! However, there are ways to speed up this process. One of them is to BSAVE the program with a "binary converter." This causes the program to be saved in tokenized form, and from then on it will load up to 12 times faster! A "binary converter" is simply an easy-to-use utility program.

There are many binary conversion programs. Almost every toolkit or utility package contains a version, and there are public domain versions, such as the CRUNCHER program on the MOAUG utility packs. We have an updated CRUNCHER from gHAAUG, and we will be distributing it as a BONUS! at the November meeting.

If you can easily find a public domain binary converter, why would you want to purchase a commercial version? Well, there are some nice enhancements to be found on commercial converters. My favorite is Intel-LOAD by Digital Express. What makes it special is that (1) it gives you two options for BSAVEing your program: one form will cause the program to be BLOADed in memory, and the other causes the program to BRUN automatically. And both options are available at the touch of a key. (2) Intel-LOAD is designed to stay in RAM. (This type of program is referred to as a "memory resident" program ; that means that you load it once, and it stays there in memory, invisible, until you turn off your machine; you can call it whenever you need it during a session.) This is a super-duper feature, especially if you have only one drive! And if you have DEI's public domain program, EZkeys II... which is a good candidate for a future BONUS... you can even call the program by touching one key! (3) The documentation is exceptionally well-done, as we have come to expect from the folks who publish Nibbles & Bits, and (4) the price is very reasonable... just $11.95 for subscribers ($15.95 for non-subscribers.) There is also a version of Intel-LOAD designed expressly for Basic 2.0.

All binary converters are descendents of the CRUNCHER program introduced by Serendipity Publications in their third (and, alas, final) issue of the Adam Technical Journal. (If you'd like to explore the nuts & bolts of BSAVEing, see this article. I may be skating on thin ice reprinting it, but I intend this as homage to its authors, who made a tremendous contribution to all ADAMites.)

"Binary converter" programs are very easy to use. When you have a program in memory that you'd like to BSAVE, just BRUN the converter. You'll see a message saying something like:

"BSAVE ----------, A49684, L3949"

Simply use the arrow keys to move the cursor under the blanks, then type in the name you wish to call your fast-loading program. (Don't put it on the same media with the same name!) Blank out any extra characters with the space bar, then move the cursor all the way to the right... past the numbers... and press <RETURN>. That's it! If you get an error message such as "Illegal Form of OS Command", ignore it. When you catalog, you will find that your program is designated as an "H" type file. When you want to run it, type in "BRUN <Filename>." Again, depending on which converter you used, you may get a message saying "File Type Mismatch." If this occurs, simply enter the command "RUN" (no need to repeat the Filename; it's in memory already at this point.)

Now, you may want to save an unconverted form of your program, because you can't read binary files from your word processor. You can, however, LIST them in Basic, once they are in memory, and print the listing if you like. Once a binary file is loaded into memory, it behaves just like any other program. You can even SAVE it as a regular Basic "A" type file to another medium.

CRUNCHER

reprinted from:

Serendipity Productions' Adam Technical Journal #3

The starter Adam system is a very good value in terms of performance per dollar. Part of the reason is the high-speed tape drive. It is many times faster than the usual cassette players used on other starter systems but is still reasonably economical. But if you use your system mostly for programming in SmartBasic, much of that speed has been compromised in order to make the most of the Smartwriter software. Most other microcomputers save a Basic program in its tokenized form just the way it sits in RAM, ready to run. Of course, doing that precludes the use of Smartwriter to edit and write Basic programs, so, instead, Adam saves the actual text that you see when you LIST the file. The price you pay for that flexibility is loading speed.

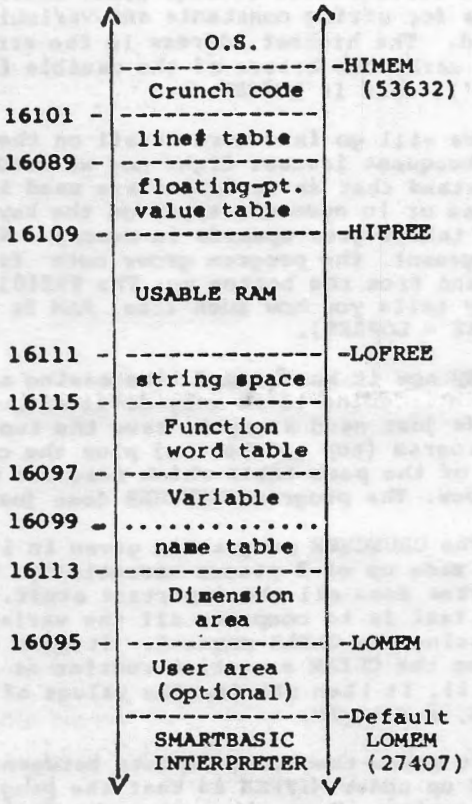
When you load a SmartBasic program, the interpreter must convert the text into tokenized code, just as if you were typing it all in That can take a lot of time. When you type in a new line in a long program or edit an existing one, you can actually see the time delay before the cursor comes back. That's pretty amazing when you consider that the CPU is running at 3 MHz. You have to execute an awful lot of machine code to produce a perceptible delay (about 35,000 instructions for an 1/8 - second delay.) This is why the tape pauses while reading a long program. It reads a block (1024 characters), then stops the tape while it tokenizes what it read, then starts reading again.

The largest program we have is a 24 block long assembler which takes 266 seconds (4 min 26 sec) to read, excluding the time it takes to locate it on the tape. Without having to tokenize anything, the Adam can read 24 blocks in 36 seconds (1.5 sec/block), using BLOAD. Wouldn't it be nice to read a Basic program from tape 7 times faster than normal? OK, so let's do it, while we learn more about the way the interpreter works!

First of all, we can't just BSAVE the tokenized program, because all the variables are stored separate from the code. Besides, since you can change HIMEM and LOMEM anytime you like, the interpreter must be saving various critical pieces of information somewhere special so it can find the program. We'll call this special place the BASIC PARAMETER TABLE, or parm table for short.

The parm table is in the middle of a rather large data area which serves a variety of purposes, but we are interested in one small piece: The 28 bytes from 16089 through 16116 that contain all the information necessary to fully map where a Basic program resides.

To understand why we need these 28 bytes, we need to know how a program is set up. Below is a pictorial map of a SmartBasic program. Along the left side are the addresses in the parm table, which define the boundaries between the various areas of the map.



A multitude of things happen when you enter a line of SmartBasic. First, the keyboard input is parsed into tokens as discussed above. Then, the tokenized line is added to the code being built below HIMEM. The lines are stored in chronological order with the first line on top and the newest line on the bottom, not in order of the line numbers.

Below the crunch code is the line number table, which is made up of 4-byte entries; two for the line number and two for the starting address of the crunch code for that line. These entries are in line number order (lowest at bottom, highest on top). This makes it easier to search through the program sequentially, such as when executing a GOTO.

Below this are the floating-point variable values (variables not ending with % or $). Only the 5-byte values are stored here. The variable names are stored in another table with pointers to the appropriate locations in this one. As lines are typed in, each area discussed so far grows downward, using up RAM as it goes, and it's all kept straight via the parm table. The value at 16109 marks the end of these tables (or the start of free RAM), so we named that value HIFREE for future reference.

At the other end of memory we have the SmartBasic interpreter. LOMEM defaults to the very top of this code, although you can set it anywhere you like, even during program execution. Above this is a 145-byte temporary area with 5-byte entries for managing dimensions, temporary values, etc.

continued, next page

CRUNCHER

continued from previous page

Above this is a two part variable-name table. This is followed by the Math Word Table described in the previous sections and, finally, by the string-value-storage area, where the text values for string constants and variables are stored. The highest address in the string-value table marks the bottom of the usuable free RAM, so we'll call it LOFREE.

We will go into more detail on these tables in subsequent issues. Right now we need to understand that as variables are used in either a program or in commands typed on the keyboard, these tables grow upwards in memory. With this arrangement, the program grows both from the top down and from the bottom up. The FRE(0) command simply tells you how much free RAM is left (HIFREE - LOFREE).

By now it must sound like saving a tokenized program is going to be very difficult, but it's not. We just need a way to save the two halves of the program (top and bottom) plus the critical 28 bytes of the parm table which describe where it all goes. The program CRUNCHER does just this.

The CRUNCHER program is given in listing #3. It is made up of 3 pieces essentially. The first 105 bytes does all the important stuff. The very first task is to compress all the variable tables by issuing the CLEAR command. It does this by calling the CLEAR execution routine as listed in Table II. It then obtains the values of HIFREE, LOFREE, and LOMEM.

It moves the block of data between LOMEM and LOFREE up under HIFREE so that the program is now one big piece. Next, it copies the Basic parms up immediately under the program. (Actually, it copies 30 bytes instead of just 28 because we like nice round numbers).

Now, we could BSAVE this piece to tape, but then we would need a second program to uncrunch this file and put the parms back so it would be a runnable Basic program. Plus we would have to tell the second program what the starting address was, since every program is of different length, and we would have to remember it everytime this file was BLOADed. Certainly not an elegant solution.

We get around this by adding an uncrunching routine (the third part of CRUNCHER) immediately under the parms. Now we're ready to BSAVE the whole thing as a binary file, but to do this we need to know the starting address for the BSAVE command. Well, register pair DE contains the address just below the start of our file as a consequence of the block-load command used to move all this stuff around. We can calculate the file length by subtracting this address from 53632 (HIMEM). All we need now is a way to communicate these values to you so you can use this in the BSAVE command.

Well, let's see. We know a bit about how the tokenized code is put together, and we know the address of the PRINT code of Table II. So why not PRINT the entire BSAVE command to the screen so all we have to do is run the cursor over it while supplying the file name, and it's done for us!!

Part 2 of CRUNCHER is just a line of crunch code to print the BSAVE command. The last task performed by part 1 of CRUNCHER is to poke the file length and start address into this tokenized code line and call the PRINT routine.

Let's quickly review the whole process again to make sure we understand it clearly. Only the first 105 bytes of CRUNCHER actually run. The next 45 bytes are the tokenized PRINT command which is read by the PRINT code and produces the BSAVE command for you to execute. The last 57 bytes is the UNCRUNCHER routine that is saved with the file for execution at load time.

So we have a Basic program in memory that we want to load fast. Just type "BRUN CRUNCHER". First it issues a CLEAR, then copies all the tables up under HIFREE, eliminating the free space in RAM. Then it copies the basic parms under that, followed by the UNCRUNCHER routine copied under the parms. The last step is to print the BSAVE command on the screen with a default file name and the appropriate length and address. You must now run the cursor over the command line, changing the name as you go, and the tokenized program is saved.

To load the program, just BRUN the saved file. It will BLOAD the file and execute the UNCRUNCHER routine. By the way, the BRUN command will always cause an error message to be displayed which insists that you have used an illegal OS command. This is another SmartBasic bug. The BRUN command works in spite of the message.

The UNCRUNCHER routine first establishes where in memory it is so it can use absolute addresses to load with. It does this by CALLing a location in high memory which is just a RETURN instruction. A machine-language CALL causes the current address to be PUSHed onto the stack so it knows where to return to (which it does immediately in this case). We decrement the stack pointer twice and POP that address into HL. Now we know the address of the first instruction after the CALL so we can calculate any other location in our routine.

The next step is to copy the parms into the Smartbasic parms table. Now, we must copy the tables down to LOMEM, but for a large program this may overwrite the UNCRUNCHER routine, causing total disaster. The way to avoid this problem is to relocate the remaining routine to a safe place, such as the trusty 56320 (more about why we chose this address at the end of this section). From the new location we can safely move the tables to LOMEM, and we're done!

We've glossed over the assembly routine fairly quickly because of its length. If you aren't familiar with Z80 code but want to understand it, we suggest buying a book on the subject such as Rodnay Zaks' "How to Program the Z80" (Radio Shack Cat. No. 62-2066). The listing is heavily commented to make it easier to follow.

We use the area of memory above 56320 quite a bit. It is a 1K buffer used by the operating system to read from, and write to, a second tape or disk file. The first file opened uses the 1K buffer immediately below 56320. Therefore, as long as we only have one file open at a time, this provides a rather large, fairly safe area of memory to work with.

FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH!

COLECO SOFTWARE AT PD PRICES!

Thanks to Rick Covell's quick thinking and the helpfulness of Lionel Playworld's manager, Kenny Vitale, we were able to score a lot of ADAM software at ridiculously low prices. When I say a lot, I don't mean we have large quantities... they didn't have LOTS of anything. If they can get hold of more, they said they'd contact us. Meanwhile, we are offering the items first to MOAUG members. Whatever's left, if anything, we will throw open to the general public. So... don't wait; if you want something, call the Bells; first come, first served. THERE MAY NEVER BE ANY MORE LOCALLY, and we beat everybody's prices nationally. Please note that this is brand new software, complete with all manuals; in the case of LOGO, CALC, and CP/M 2.2, the manuals alone are easily worth the price. Some of this stuff is on disk. As before, we will copy it to tape for you if necessary.

PRICES: (These prices INCLUDE tax!)
$5.50: Flashcard Maker, Flashcard Trivia, SmartFiler, Recipe Filer (ddp)
$6.50: Flashbacks, Vocabulator, Recipe Filer (disk)
$7.50: Best of Broderbund (AE/Choplifter), Family Feud, Scarry's Word Book
$10.50: ExperType, SmartLOGO, ADAMCalc
$12.50: Smart Letters & Forms, CP/M 2.2

Note: Recipe Filer must be disk to operate on disk. Flashbacks, Vocabulator and Trivia require Flashcard Maker. Those are the only restrictions.

As an additional bonus for members who order before the next meeting, WE WILL GIVE YOU AN ADDITIONAL 10 PER CENT OFF YOUR ORDER. 10% OFFER IS NOT LIMITED TO THOSE WHO ATTEND THE MEETING, BUT IT IS LIMITED TO THOSE WHO ORDER NEXT WEEK. THIS OFFER WILL NEVER BE REPEATED. Supplies of some items are VERY limited.

CALL ALLEN & FRANCES BELL AT 352-0724. REPEAT: FIRST COME, FIRST SERVED!!!

FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH! FLASH!

Big John of Orphanware has made us a special offer. Simply return the coupon below to take advantage of the lowest price in the country on new 64k cards, from the hottest hardware developer in the country... folks you can trust!!!

CLIP & SEND
MOAUG MEMBERSHIP SPECIAL!
TO: ORPHANWARE P.O. BOX 324 CANAL FULTON, OH 44614

YES! PLEASE SHIP ME A BRAND-NEW MX-64 MEMORY EXPANDER FOR ONLY $23.95 PLUS $3.00 SHIPPING. I ENCLOSE
☐ CHECK ☐ MONEY ORDER

NAME _____
ADDRESS _____
_____

EDITOR: PATRICIA HERRINGTON
1003 OAK LANE
APOPKA, FL. 32703

ADAM LIVES!