NOTES ON SMARTBASIC 2.0

(1) SmartBASIC II occupies 49 blocks on the disk, in contrast to SB I's mere 28 K. The file size is so unexpestedly 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 Lagerquist, author of the revisions to SmartBASIC that were incorporated into V2.0, this revision comes equipped with all the fixes to the file-handling functions that enable it to work "much faster" with data files opened by BASIC. SB II is apparently capable of correctly opening and read/writing random-acess 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 new and includes a RAM test for the presence of the 64-K Memory Expander. Also, SB II will boot from ANY drive and set that as the active drive, 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 is 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 1594 in SB II, and at 16095 in SB I. 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 gets an "Out of Memory Error."

(4) Both REM and DATA have been fixed for the bug in SB I that keeps inserting a space after them on a statement line each time a program is SAVEd or LOADed. In fact, SB II specifically strips out any extra spaces with each listing. THIS SINGLE FEATURE MAKES SB II WORTH HAVING, JUST TO BE ABLE TO SAVE PROGRAMS WITH A MINIMUM OF FILE SPACE.

(5) The cursor-display routine has been rewritten, so now the underline cursor 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 is now accompanied by a "beep" (CHR$(7)).

(7) Four of the special-function keys are now 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). However, this screen dump is now INTERRUPTABLE WITH A CONTROL-C; if the CONTROL-C is issued during a screen dump from within a program (PRINT CHR$(16)), control returns to the next step without a "Break." The Printer also Prints any INVERSE CHARACTERS in the text that have normal characters on the daisey wheel, as NORMAL CHARACTERS. Also the Printer can be made to print backwards, as with Basic 1.0, with CHR$(15). This feature was useless however on Basic 1.0 since the printer would lock up and continue printing the same 80 characters forever. This bug has been fixed in Basic 2.0 and the Printer will stop and return control to the program.

(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 routine that involves updating the spinner (the rotary device 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 NONFUNCTIONAL as in SB I.

(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 an 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 projects.

(12) The INIT command now 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 "CATALOG" shows an impossible "253 Blocks Free"!

(13) There is virtually NO imposed POKE LIMIT in Basic 2.0, (you can poke up to FEBF {65215}; POKEs are rejected above that limit only to prevent poking into the DCB's, which could be catastrophic). The old pokes to 16149 and 16150, used with Basic 1.0 to re-set its Poke Limit SHOULD NOT BE USED with Basic 2.0. These are in a different part of the Basic 2.0 machine language and will blow-up the Basic Boot and require re-booting. This is true with the old Basic 1.0 Color Pokes as well. DON'T use a program written for Basic 1.0 that changes the Screen colors. These programs must be modified first. The new color pokes are:

```
     POKE 17184 <0-15> for the Border
     POKE 17240 <Normal Text *16 + Screen>
     POKE 17251 <Inverse Text*16 + Inverse Background>
```

(14) The commands STDMEM and EXTMEM have been added to SB II to permit switching between standard memory (64 K RAM plus 16 K VRAM; also the default configuration) and extended memory, which uses the 64 K Memory Expander (if present) in much the same manner as SmartWriter uses the extra RAM to expand its workspace, i.e., through a bank-switching arrangement. The command to load the bank-switched version (which takes up some of that extra space in the SB II disk file) is "EXTMEM", which must be entered 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 (or 90656 after "NEW"). Compare this with the value of 26391 (or 26401 after "NEW") in SB II's standard memory configuration. To return to 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 any error checking in EXTMEM, and entering an illegal value can cause an immediate system crash! Also, programs will generally run more slowly (about half as fast) in EXTMEM than is STDMEM, because of the extra CPU time spent in bank switching. That's the way it goes!

(15) High-resolution graphics are suppossedly fixed 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 the address POKEd into these two bytes. There are TWO default sprites already prvided; 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 you own sprites is EXACTLY THE SAME AS FOR SHAPES IS SMARTLOGO, as described in the 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.

(16) 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 funciton, 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 screen diplay of the entire Adam character set, from ASCII (a centered dot) through 127 (a flashing cursor).

(17) There is another change that really IS an improvement, but takes some getting used to. When editing previously writen lines, if you should make an Error, The ERROR is ACCEPTED, and the Original Line is LOST. The line is then printed on the screen with the "Sad Face" character preceeding it. It must then be re-written. "Sad Face" lines will stay in the listing untill changed, but will Break the program when Run, if between sequential lines. This differs considerably from Basic 1.0, when any errors in editing were REJECTED, and the original line was returned UNCHANGED. I've found it good insurance to copy lines before editing them by adding 9000 or some such to them. This then gives me an unchanged line to fall back on if I make an error and lose the line.