NIBBLES & BITS

THE COMPREHENSIVE MONTHLY NEWSLETTER FOR ADAM USERS

Published by DIGITAL EXPRESS, INC.
1203 Northwoods Drive, Kings Mtn., NC 28086
(704) 739-9106

AUGust 1986
vol: 1, nmb: 2
SINGLE ISSUE: $3.50

INSIDE THIS ISSUE:

This issue includes 15 SmartBASIC program LISTs, 5 tables (charts), and 6 assembly-language lists.

EDITOR'S NOTE

As this issue (our second one) goes to press, we ALREADY have one of the largest circulations of any ADAM newsletter. We owe this success to you, our fellow ADAMites. THANK YOU!!!

A great many of you have mailed in your questions and comments. In response, we're adding several new departments to the newsletter. We are committed to providing you with the ADAM information (tutorial, general, etc.) that you want to read.

Coleco's decision to drop the ADAM computer has induced a fervent unity among active ADAM users around the world. We ADAM users stay in touch with one another through local users groups, international users groups, computer bulletin boards, and newsletter publications.

You have fellow NIBBLES & BITS subscribers throughout the USA, in Puerto Rico, in Canada, and in Great Britain. We are very proud to be a part of this INTERNATIONAL TEAM OF ADAM USERS.

N&B NEWS

→ Many of you have asked us to describe our system configuration that creates this newsletter. The newsletter is printed with a Panasonic KX-1091. This dot-matrix printer is interfaced to ADAM via the EVE SP-1. Our design editor uses an unreleased DATA DOCTOR software package which converts standard SmartWriter files to our newsletter format. This software had no documentation, and we are still discovering some of its features. This newsletter reflects a few of our latest discoveries.

→ New dapartments in this issue are: EXPANDING YOUR SYSTEM and N&B BULLETIN BOARD. Next month we're starting our CP/M™ 2.2 workshop and the ADAM USERS' FORUM.

→ We are currently working on several topics for the HACKER'S DELIGHT department. We'll continue our study of the VDP including how to save TEXT and HGR screens and how to use the video chip to increase the length of your BASIC programs (even without the 64K expander), how to design your own fonts without employing any standard BASIC RAM, and how to create and use sprites. We'll take a look at hi-res shape tables and bit-mapping. We'll study the BASIC interpreter in detail and fully explain storage media configurations. One of the most intriguing of the upcoming topics will be a discussion on how to create a ramdisk in BASIC (with and without the 64K expander). If you don't have Intel-LOAD or one of the other popular binary BASIC program converters yet, get it now as we'll use one of these to create ramdisks.

→ The DEI selection of ADAM products is growing. We will try to carry all the software that you rate in the SWIFT POLL.

→ Until 12-1-86 NIBBLES & BITS will offer the following bonus to YOU for having a friend subscribe to our newsletter. For each friend that you have subscribe, we will extend your subscription by one issue (limit 12). If you get 12 friends to subscribe, you'll get an entire extra year of NIBBLES & BITS absolutely FREE. Just have them send their check or money order with a letter including your N&B ID number and your name.

→ There were a couple of typos in the last issue. On page 11, the HGR TEXT color change should have read: "POKE 25568, (nl * 16) + ns". On page 18 and on the PRODUCT LIST the "over 700 documented op codes" should have read: "approximately 700 documented op codes".

→ Please include your subscription ID number on all correspondence to us. If you want a personal reply, just send an SASE.

→ Your subscription ID number is on the first line of your mailing label affixed to the newsletter. It is a ten digit code. The first four digits are the month and year of the last N&B issue in your current subscription. Following the ID code is a brief message. If this is your final issue, the message will read "FINAL ISSUE!!!". If this is your penultimate issue, the message will read: "PLEASE RENEW NOW". Otherwise, the message will apprise you of the exact number of issues remaining in your subscription. Please verify this info each month.

EDITOR-IN-CHIEF:
Vernon L. Whitman, Sr. (Luke)
TECHNICAL CONSULTANT:
Dr. Solomon Swift
DESIGN EDITOR:
Tim Whetstine
CONTRIBUTING EDITORS:
Jeff Smithers
Janet Weston
Ted Johnson

ADAM NEWS

→ EVE Electronics has another first! As you read this newsletter, EVE should have just completed production of their own ADAM compatible disk drives! They have the standard 160K single-sided 5 1/4 inch drive (about $230) and a new 320K double-sided drive (about $310). They will also upgrade standard ADAM disk drives to double sided for about $150. And they will repair ADAM disk drives for about $40 plus parts.

→ DEI will soon release their third machine code software package, IntelFONTS. This package includes four alternate TEXT font sets. Each one can be used in TEXT, GR, or HGR. You can even use different INVERSE and NORMAL fonts. And, the sets don't use any standard BASIC RAM. The package also includes a shape table with 118 font shapes for use in HGR graphics.

→ Last month we mentioned American Design Components as a supplier of ADAM standard system spare parts. We have others listed in this month's N&B BULLETIN BOARD. These electronic parts warehouses purchased Coleco inventory close-outs in bulk quantities. We strongly recommend that you get some of these spare parts as backups.

→ If you're a CP/M hacker, try NIAD's public domain library. They have eighteen libraries -- most contain over 130K of assorted programs. This service is a great time-saver and the cost per volume is very low to NIAD members.

→ If you still haven't sent in the Programmer Survey in the last issue of <u>FAMILY COMPUTING</u> (page 49 - July, 1986), please do so. And, please let them know that you missed the ADAM programs in their July issue.

→ The ADAM-X-CHANGE is sending out their newsletter again. If you don't access their COBRA BBS, you can get info about the newsletter from the following address:

Robert Wright
RFD #one, Box 528
Saco, ME 04072

→ As mentioned in the last newsletter, DEI has purchased the copyrights to all DATA DOCTOR software. If you have a matter of concern for that company, please address it to:

DIGITAL EXPRESS, INC.
ATTN: SS
1203 Northwoods Drive
Kings Mtn., NC 28086

→ Many ADAM owners already have SmartBASIC 2.0. This package has not been released by Coleco or Lazer Microsystems. Neither company acknowledges copyright ownership. Whether or not distribution of this program is software piracy, is a very moot point. Also, there are several different versions in circulation.

EXPANDING YOUR SYSTEM

This new department is intended to help you get a better perspective on expanding your standard system. Topics will be discussed in general terms. We'll start with the basics progressing toward advanced peripheral applications.

COPY UTILITIES:

The very first software addition to your ADAM should be a copy utility. These programs written partially or entirely in machine language are used to make backups of your software. When you backup a medium, you should store the original away in a safe place using only the backup. Backing up software is a safeguard that you can not afford to overlook.

Copy utilities work by reading blocks of data from a source storage medium, storing them temporarily in a RAM buffer, and then transferring the data to the destination storage medium. You only need one drive to use a copy utility. However, the process is much less user involved when you employ two drives.

Coleco released two formats of data packs. You may encounter a problem in the copy process, if you don't use the same format for both the source and the destination. To distinguish the formats: (1) turn ADAM on, (2) if it was already on, turn it off first, (3) insert the data pack and use SmartWriter to get a directory, (4) if the tape is centered between the spindles, it's a center format, or (5) if the tape is mostly on the left spindle, its a right directory format.

The quality, features, and price of copy utilities vary greatly. There are several written purely in machine code which include over a dozen extra features. These typically sell for about thirty dollars. Some are written primarily in BASIC. These usually sell for about twenty dollars. There are exceptions to these price trends. For example, one company sells a machine code copier with many features and extensive documentation for under FIFTEEN dollars. Another company, sells a BASIC copier with NO extra features for OVER FORTY dollars. Also, there are several BASIC media copiers in the public domain (FREE). Probably the best of these is UTILCOPY by Wayne Motel of NIAD.

Copy utilities are intended for archival (personal library) use ONLY!!! Do NOT trade backups for other software. This is illegal and can be highly detrimental to ADAM's future.

BIT BY BIT

Common Questions:

What is meant by "bits" and "bytes"?

Each chip in a computer contains many thousands of storage cells. Each cell is called a bit (BInary digiT). A bit is the smallest unit of information that a computer deals with. Each bit is either an "ON" or an "OFF" in an electrical circuit. Mathematically, we refer to these two states as either a "ONE" or a "ZERO". Thus, the value of bits can be expressed as a binary (two-value) system.

A byte is the number of bits necessary to represent the largest element of data transfer within the computer. With ADAM, a byte is equivalent to eight bits. Specific bytes of memory are called "addresses" or "locations".

What are ROM and RAM?

These two terms refer to specific types of memory chips within a computer. RAM (Random Access Memory) is the programmable part of memory.It is erased every time the power is turned off. This is the memory in which your programs are stored and executed.

ROM (Read-Only Memory), on the other hand, can never be programmed or changed. However, once the information from the ROM chip (such as ADAM's Operating System) is read into RAM, it can be changed.

What is a ramdisk?

A ramdisk is a technique that allows you to have more than one program in memory at any given time. These programs may be BASIC, machine code, etc.

Data Types:

SmartBASIC permits two types of data: string and numeric. Each of these data types is handled differently by ADAM.

A string is one or more ASCII characters. Strings are generally considered to be words. However, a string may also contain numbers.

SmartBASIC also permits a string of zero characters. This is called a "null string". It is analogous to a value of zero in mathematics.

Numeric data is information that is handled as numbers. Mathematic operations may be performed on numeric data. SmartBASIC permits three classes of numeric data: fixed-point, floating point, and integer.

Fixed-point numbers are the most common. This type of number may contain a decimal portion. Examples of fixed-point numbers are:

42100
.00234
566.43

Floating point numbers are represented in scientific notation. SmartBASIC will automatically convert a fixed point number to floating point if it is too large or too small. Examples of floating point numbers are:

4.21E+4
2.34E-3
5.6643E+2

Integers are a specific type of fixed-point numbers. An integer contains no decimal portion. And an integer may be in the range of -32767 to 32767, inclusive.

Examples of acceptable integers are:

```
32767
-4592
563
```

In scientific notation the "E" represents a power of ten. For example:

```
2.5E+3 = 2.5 * 1000 = 2500
3.7E-2 = 3.7 * .01 = .037
```

Variables:

A variable is a term that describes an area of memory that is represented by a name. SmartBASIC automatically assigns an area of memory for each variable so that you do not have remember where the variable's value is stored.

The name of a variable may be virtually any character length. However, ADAM only recognizes the first two characters. The first character of a variable name must be an alphabetic letter. And a variable name must not duplicate one of BASIC's command words.

To differentiate the data types and data classes with variables, you need to follow the variable name with a specific suffix. Strings end with a dollar sign ($). Integers end with percent sign (%). Fixed-point and floating point numbers do not require a variable name suffix. SmartBASIC does not allow a string value to be assigned to a numeric variable or vice versa.

SmartBASIC initially assigns a value of zero to numeric variables and the null string value to string variables. Values may be assigned to a variable as the result of a calculation or as the result of programmer assignment.

The LET command:

The SmartBASIC command "LET" allows the programmer to assign a value to a variable. This command is used so frequently that a shortcut for it is standard in nearly every version of BASIC. The shortcut is to simply not type the word LET; rather, it is understood. For example,

```
x = 10
```

is the same as

```
LET x = 10
```

With the LET command the value on the right of the equals sign is assigned to the variable name on the left of the equals sign. The value assigned to a variable may be a constant, another variable, or the result of an operation. For example:

```
name$ = "Alfred"
last$ = "Smith"
full$ = name$ + " " + last$
PRINT full$
```

Don't confuse algebraic equality with variable value assignments. For example the following assignment would be absurd as an algebraic equation.

```
y = y + 2
```

The CLEAR command:

The SmartBASIC command "CLEAR" has the opposite effect of the LET command. CLEAR sets all numeric variables to zero and all string variables to null.

BYTE-SIZED BASIC

The ASCII Code (PART II):

This month we'll concentrate on the ASCII code with respect to generating true random numbers using the keyboard input buffer. To begin with, let's take a look at the RND command.

The RND(x) function will return a number greater than zero and less then one, ie, a decimal number (usually 9 digits to the right of the decimal). It is sometimes convenient to convert this decimal value to an integer within a specific range. The formula is:

```
x = INT((z - y +1)*RND(1)) + y
```

This equation will return a number greater than or equal to "y" and less than or equal to "z". For example, if you want a random number within the range of 25 and 100 inclusive, you would substitute these values in the equation as follows:

```
x = INT((100-25+1)*RND(1)) + 25
```

or,

```
x = INT(76*RND(1))+25
```

RND has a limited use as it is designed by SmartBASIC. Every time a program is RUN, the same sequence of random numbers is generated. The program at the top of page 8 demonstrates this problem. Every time you RUN the program, the same random (???) number is printed.

You can overcome this RND limitation if you employ a technique called "negative seeding". This process involves using a negative argument with RND, eg, y = RND(-x). Thereafter a new sequence of random numbers is generated. Some simplistic BASIC programs seed the RNG (Random Number Generator) with the negative of a user input number. The second program on page 8 demonstrates this algorithm. The drawback here is that there is a particular sequence of random numbers that corresponds to each negative seed, ie, every time the user inputs the same number, the same random numbers will follow.

The easiest method to attain true randomization in BASIC is to have the user seed the RNG oblivious to your technique. The third program on page 8 illustrates this method. Using the GET or INPUT command stops program execution and waits for user response. By reading the keyboard input buffer, however, program activity can contiunue irrepective of user activity.

This program uses a loop that checks to see if a key has been pressed with each pass. If not, the counter is incremented by one. When a key is pressed the RNG is seeded with the negative of the current counter value. Thus, you have access to true randomization provided the user doesn't press a key at precisely (to the one-hundreth of a second) the same time interval with each program execution.

Note that this program starts off by POKEing a zero into the input buffer. To POKE up that high, you need to set to POKE limit to 65535, ie, POKE 16149, 255 and POKE 16150, 255 (255*256 + 255 = 65535).

The final program on page 8 shows a similar trick. This one uses the game controller.

```
 10 REM BASIC RND bug demo
 20 nu = INT(RND(1)*5000)+1
 30 PRINT nu
```

```
 10 REM user seeding RND demo
 20 INPUT " enter any positive number: ";nb
 30 x = RND(-nb)
 40 nu = INT(RND(1)*5000)+1
 50 PRINT nu
```

```
 10 REM input buffer seed RND demo
100 TEXT: PRINT " press any letter key . . ."
110 POKE 16149,255: POKE 16150,255: POKE 64885,0
120 counter = 1
130 IF PEEK(64885) <> 0 GOTO 200
140 counter = counter+1
150 IF counter > 30000 THEN  counter = 1
160 GOTO 130
200 x = RND(-counter)
210 nu = INT(RND(1)*5000)+1
220 PRINT " ";nu
```

```
 10 REM RND demo using game controller
100 TEXT: PRINT " press the right fire button:"
110 counter = 1
120 IF PDL(9) = 1 GOTO 200
130 counter = counter+1
140 IF counter > 30000 THEN  counter = 1
150 GOTO 120
200 x = RND(-counter)
210 nu = INT(RND(1)*5000)+1
220 PRINT " ";nu
```

Simple Tricks:

The first program on page 10 illustrates a simple average program. The program continues in a loop until you enter a negative number.

The INT command makes it easy to get rid of the decimal part of a number. But, how do you get rid of the integer and keep the decimal? The next program on page 10 shows you how.

The third program demonstrates how to determine if a number is odd or even. One use of this algorithm is illustrated in the final program on page 10. This one will automatically center a word on the screen. It uses INVERSE for uniform centering of both odd and even length words.

POKEs To Play With (PART II):

The FLASH Command:

Address 159 controls the speed at which the FLASHing characters blink. It's default value is 12. One is the fastest speed and 255 is the slowest.

Address 17006 controls FLASH. If you POKE a 128 into 17006, you'll activate FLASH. A zero will turn it off. The first program on page 11 shows you how to FLASH between upper and lower case letters rather than between NORMAL and INVERSE letters. INVERSE fonts have an ASCII value of 128 higher than their corresponding NORMAL fonts. Lower case fonts are 32 higher than their corresponding upper case fonts.

BASIC Interpreter Routines:

CALLing the following machine code routines (within the interpreter's area of RAM) is the exact equivalent of using the corresponding BASIC commands.

```
CALL  8141 = CLEAR
CALL  8109 = CLRERR
CALL  6387 = CONT
CALL  6047 = END
CALL 11050 = FLASH
CALL 11070 = GR
CALL 11075 = HGR
CALL 11080 = HGR2
CALL 11090 = HOME
CALL 11055 = INVERSE
CALL  7407 = LIST
CALL  6356 = NEW
CALL 11060 = TEXT
CALL  6341 = NOTRACE
CALL  8493 = POP
CALL  9482 = RESTORE
CALL  8313 = RESUME
CALL  8477 = RETURN
CALL  6169 = RUN
CALL  6378 = STOP
CALL 11065 = TEXT
CALL  6336 = TRACE
```

Some CALLs have no SmartBASIC equivalent. CALL 64743 is an instant soft switch to SmartWriter. CALL 64809 will instantly remove all sprites from the screen without disturbing anything else. CALL 64851 instantly turns off all voices and the noise generator. And, CALL 16588 is roughly the same as RUN HELLO.

These equivalences also apply:

```
CNTL-S = POKE 16136, 0
COLOR  = POKE 16776, (0 - 15)
FLASH  = POKE 17006, 128
HCOLOR = POKE 16777, (0 - 15)
SCALE  = POKE 16765, (0 - 255)
SPEED  = POKE 16129, (0 - 255)
```

```
  10 REM simple average demo
  50 TEXT
 100 PRINT " Enter positive numbers to "
 110 PRINT " average; enter a negative "
 120 PRINT " number to end.": PRINT: PRINT
 130 count = 0: total = 0
 200 INPUT " number please: ";nu
 210 IF nu < 0 THEN  PRINT " that's all!!!": END
 300 count = count+1: total = total+nu
 310 avg = total/count
 320 PRINT " entered: ";count;" numbers"
 330 PRINT " average: ";avg: PRINT: PRINT
 340 GOTO 200
```

```
  10 REM integer dump demo
  50 TEXT
 100 PRINT " Enter a number with several "
 110 PRINT " digits to the right of the "
 120 INPUT " decimal: ";nb: PRINT: PRINT
 200 de = nb-INT(nb)
 210 PRINT " The decimal portion is: ";de
```

```
  10 REM odd or even demo
 100 INPUT " Enter a number: ";nu
 110 IF INT(nu/2) = nu/2 THEN  ans$ = "even": GOTO 130
 120 ans$ = "odd"
 130 PRINT " ";nu;" is ";ans$;"."
```

```
  10 REM auto word center routine
  50 TEXT
 100 INPUT " enter a short word? ";word$
 110 TEXT: GOSUB 1000: PRINT: PRINT: PRINT
 120 INPUT " enter another short word: ";word$
 130 GOSUB 1000
 200 END
1000 word$ = " "+word$: lw = LEN(word$)
1010 IF lw/2 = INT(lw/2) THEN  word$ = word$+" ": lw = lw+1
1020 INVERSE: HTAB 16-lw/2: PRINT word$: NORMAL: RETURN
```

```
   10 REM playing with FLASH #1
  100 TEXT: POKE 17006,32
  110 PRINT " ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  120 NORMAL: PRINT: PRINT
  130 FLASH: PRINT " ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  140 NORMAL: PRINT: PRINT
  150 PRINT " How about that!!"
```

```
   10 REM routine to temporarily save a GR screen
   20 DIM grsaver(39,39)
  100 REM replace this simple box with your own GR screen
  500 GR: COLOR  = 14: FOR x = 0 TO 39 STEP 2
  510 HLIN 0,39 AT x: NEXT
  520 VLIN 5,15 AT 15: VLIN 5,15 AT 25
  530 COLOR  = 7: HLIN 15,25 AT 5: HLIN 15,25 AT 15
  540 VLIN 5,15 AT 15: VLIN 5,15 AT 25
 1000 GOSUB 30000: REM temporary screen save
 1010 GR: PRINT " enter GOSUB 40000 [RETURN], "
 1020 PRINT " to bring it back.": END
30000 PRINT " standby: saving screen"
30010 FOR x = 0 TO 39: FOR y = 0 TO 39
30020 grsaver(x,y) = SCRN(x,y): NEXT: NEXT: RETURN
40000 GR: PRINT " restoring saved screen."
40010 FOR x = 0 TO 39: FOR y = 0 TO 39
40020 COLOR  = grsaver(x,y)
40030 PLOT x,y: NEXT: NEXT: RETURN
```

AMAZING
DYNAMIC
ADAPTIVE
MICRDCOMPUTER

The Cursor:

Address 16953 contains the ASCII value of the cursor. Its default value is 95, ie, an underscore. You can eliminate the cursor from the screen with a "0" or a "32".

Address 17001 contains the current vertical position of the cursor. Address 17002 contains the current horizontal position of the cursor.

You can stop the cursor from blinking by POKEing a "1" into address 17000. A zero or TEXT will restart the blinking.

Address 17291 controls the speed of cursor blinking. The default value is 4. One is the fastest and 255 is the slowest.

The Prompt:

Address 1146 contains the ASCII value of the prompt. Its default value is 93, ie, a right bracket. Address 1145 contains the number of prompt fonts. Address 1147 contains the ASCII value of the second prompt font.

The Built-in Bell:

PRINT CHR$(7) will sound BASIC's built-in bell. You can change its parameters. The duration is at address 17963. The volume is at address 17958. And you can change the tone with addresses 17950 and 17954.

■■■■■■■

How To SAVE A GR Screen (PART I):

This month we'll start a short series on how to save your GR screens. Next month we'll present a rather long program that will allow you draw, change colors, erase, save to tape or disk, and load from tape or disk.

The second program on page 11 illustrates a technique for temporarily saving your GR screens. This same principle is used in next month's program. For now though, study the algorithm logic until you feel comfortable with it.

Line numbers 500 through 540 create a very simple GR drawing. The subroutine at line 30000 temporarily saves the screen. And the subroutine at line 40000 restores the screen. With the save routine, the double-dimensioned variable "grsaver" stores each horizontal and vertical color coordinate from the SCRN command.

With the restore routine, the reverse procedure takes place. Here, COLOR is set to the value of a screen coordinate. Then the coordinate is PLOTed. The process continues until each GR block is PLOTed. You should note that the restored screen will be an exact replica of the saved screen. However, the sequence of drawing will be different.

■■■■■■■

HACKER'S DELIGHT

Hacking In Perspective:

Hacking encompasses quite a variety of computing aspects. Chief among these is the ability to program competently. As an advanced BASIC programmer, you are no doubt frustratingly aware of the language's limitations. Machine code programming, on the other hand, presents an immensely greater creative latitude. In fact, as a proficient machine code programmer you are restricted only by your creativity and the peripherals attached to your system.

The goal of this department is to help you get the maximum benefit of your ADAM through independent software design. By learning in a logical step-by-step process, you'll learn faster and more thoroughly. The first issue of N&B covered a few of the very elementary concepts involved with machine coding. This month we assume that you understand each of those topics and that you are READY FOR MORE. After we experiment with several of the OS routines, we'll start developing entire programs in machine code. One of our first will be "DiskMASTER", a media copy utility with several sophisticated features. But for now, let's continue to lay a solid foundation in the basics . . .

Assembly Language Notes:

→ Machine coding is called the "object code" of a program. Any higher level representation of the object code is referred to as the program's "source code".

→ An assembly-language source statement may consist of two components: the op-code (the mnemonic) and then (depending on the operation) the operand. The operand represents the entity upon which the operation is performed.

→ For certain operations, the ensuing operand must contain two parts separated by a comma. In this event, the operand on the left is called the destination. And, the operand on the right of the comma is called the source. This is logical terminology as data is passed from the source operand to the destination operand.

→ In decimal format, the low order byte precedes the high order byte. In hex format, the low order byte follows the high order byte.

→ In most assembly-lnaguage lists, hex values are denoted by a preceding dollar sign ($). In some cases, however, hex values are indicated by an ensuing letter "H".

Video Display Modes:

ADAM's video chip is capable of three video display modes. These are not TEXT, GR, and HGR.

The modes are: 32-column text mode, 40-column text mode, and multicolor mode. Each mode can be used with split-screen graphics enabled (GR and HGR) or disabled.

BASIC TEXT mode is the 32-column video mode. This mode permits 256 individual characters. Each character has a potential dimension of 8-by-8 pixels (PIcture ELements - screen dots). Beginning with the first character, color can be set for every eight characters. The screen image table is 768 (32 * 24) bytes in length.

The 40-column video mode also supports 256 characters. These characters have a potential dimension of 6-by-8 pixels. All characters must be one color. This color is determined by video register 7 -- (nl * 16) + bs. The screen image table is 960 (40 * 24) bytes in length. This mode does NOT support sprites.

Multicolor mode defines each character as a 4-by-4 pixel block. This allows for a screen image of 3072 (64 * 48) bytes.

To change video display modes it is necessary to configure the eight bits of video register one accordingly.

Bit Configurations:

Each byte consists of eight bits. When a bit has a value of one, it is described as being "set". When a bit has a value of zero, it is described as being "reset".

Each set bit is assigned a value that is a power of two. The eight bits are numbered sequentially from the rightmost bit (7, 6, 5, 4, 3, 2, 1, 0).

If a byte only has the leftmost bit set (#7), then the byte's value is 2^7 or 128. If no bits are set, the byte value is zero. If all bits are set, the byte value is 255.

Video Register One:

The controls of video register one extend beyond the selection of a video display mode. The following table describes each of the register's bits.

BIT #7 (set value = 128 or 2^7)
set: 16K VRAM enable
reset: 4K VRAM enable

BIT #6 (set value = 64 or 2^6)
set: display enable
reset: display disable

BIT #5 (set value = 32 or 2^5)
set: interrupt enable
reset: interrupt disable

BIT #4 (set value = 16 or 2^4)
set: 40-column text
reset: not 40-column mode

BIT #3 (set value = 8 or 2^3)
set: multicolor mode
reset: not multicolor mode

BIT #2 (set value = 4 or 2^2)
set: no effect
reset: no effect

BIT #1 (set value = 2 or 2^1)
set: 8-by-8 sprite enable
reset: 16-by-16 sprite enable

BIT #0 (set value = 1 or 2^0)
set: normal sprite size
reset: double magnification

Both bit#3 and bit#2 must be RESET to achieve 32-column mode. You can NOT SET BOTH bit#3 and bit#2. Also, you should always set the top 3 bits (#7, #6, and #5).

The VDP Programs:

The program on the top of page 16 illustrates the three video display modes. As you'll notice SmartBASIC is not set up for 40-column display. To correct this requires several POKEs (we are still working on them). However, the 40-column mode works very nicely in pure machine code programs. The first assembly-language list on page 17 details the machine code set up for changing video display modes.

You can also change modes with two BASIC POKEs. POKE 17059 with the background and font color combination and POKE 17215 with the video mode value.

The second program on page 16 shows you how to PEEK and POKE VRAM. The machine coding is explained in the last two assembly-language lists on page 17.

These two OS routines are designed for transferring entire blocks between VRAM and RAM. We have set the byte count to "one" for our current purpose. However, we'll elaborate on transferring data tables in a later issue.

The Directory (PART I):

The directory of a storage medium (disk or data pack) is typically contained in block one. The BASIC CATALOG command reveals some of the directory's information. Each file in the directory is assigned a 26-byte slot. This slot contains various data regarding the file (name, length, etc.).

In a one block directory (the standard length) these 26-byte slots allow for 39 file enties. Four of these are system allocated. These are volume, BOOT, DIRECTORY, and BLOCKS LEFT. All user files are inserted between the DIRECTORY slot and the BLOCKS LEFT slot.

By increasing the block length of the directory you can store more than 35 files on a medium. A two block directory permits 74 user files. A three block directory permits 114 user files. ADAM can behave unpredictably if you try to extend the directory size beyond three blocks.

By POKEing values in the correct addresses, you can modify INIT to suit your needs. Let's take a look at how INIT works.

ADAM loads the current drive code from address 16821. The values are: tape one = 8, tape two = 24, disk one = 4, and disk two = 5. You can change the drive by POKEing the coded value into that address or you can you use BASIC's "d" suffix.

The maximum length for the volume name is at address 23328. The default value is 10. You can set it to 11, but you'll need to change it back to ten when you're done INITing.

The name you select is first stored in a buffer (temporary storage area) starting at address 16797. BASIC terminates the name with a three (this is ASCII code for end of string).

BASIC checks to see if the medium contains the file BASICPGM (Smart-BASIC). If it does, INIT is aborted. This string is at address 25257.

(continued on page 21)

```
  10 REM video mode demo
  50 LOMEM :28000: TEXT
 100 DATA 6,1,14,192,205,32,253,6,7,14,23,205,32,253,201
 110 FOR x = 27600 TO 27614: READ ml: POKE x,ml: NEXT
 200 PRINT " Which option: ": PRINT: PRINT
 210 PRINT " 1 = 32 column text": PRINT " 2 = 40 column text"
 220 PRINT " 3 = multicolor mode": PRINT " 4 = exit program"
 300 GET key$: IF key$ < "1" OR key$ > "4" THEN  RUN
 310 IF key$ = "1" THEN  pk = 192
 320 IF key$ = "2" THEN  pk = 240
 330 IF key$ = "3" THEN  pk = 232
 340 IF key$ = "4" GOTO 400
 350 POKE 27603,pk: CALL 27600: GOTO 300
 400 TEXT: PRINT " program terminated.": END
```

```
  10 REM VRAM PEEK and POKE
  50 LOMEM :28000: TEXT
 100 DATA 17,0,0,33,255,107,1,1,0,205,26,253,201
 110 DATA 17,0,0,33,255,107,1,1,0,205,29,253,201
 120 FOR x = 27600 TO 27612: READ po: POKE x,po: NEXT
 130 FOR x = 27613 TO 27625: READ pe: POKE x,pe: NEXT
 200 HOME: PRINT " VRAM PEEK and POKE": PRINT
 210 PRINT " 1 = VRAM POKE": PRINT " 2 = VRAM PEEK"
 220 PRINT " 3 = exit program"
 230 GET key$: IF key$ < "1" OR key$ > "2" GOTO 9000
 240 IF key$ = "2" GOTO 400
 300 PRINT: INPUT " Which VRAM address to POKE? ";po$
 310 po = VAL(po$): IF po < 0 OR po > 16383 GOTO 300
 320 PRINT: INPUT " What value to POKE? ";pv$
 330 pv = VAL(pv$): IF pv < 0 OR pv > 255 GOTO 320
 340 ph% = po/256: pl% = po-ph%*256: POKE 27647,pv
 350 POKE 27601,pl%: POKE 27602,ph%: CALL 27600
 360 PRINT: PRINT " press any key . . ."
 370 GET key$: GOTO 200
 400 PRINT: INPUT " Which VRAM address to PEEK?";pe$
 410 pe = VAL(pe$): IF pe < 0 OR pe > 16383 GOTO 400
 420 ph% = pe/256: pl% = pe-ph%*256
 430 POKE 27614,pl%: POKE 27615,ph%: CALL 27613
 440 PRINT: PRINT " The value at ";pe$;" is: ";PEEK(27647)
 450 IF PEEK(27647) < 32 OR PEEK(27647) > 126 GOTO 470
 460 PRINT " The character is: ";CHR$(PEEK(27647))
 470 PRINT: PRINT " press any key . . ."
 480 GET key$: GOTO 200
9000 TEXT: PRINT " program terminated.": END
```

TITLE (asmb#4): Video Mode Changer

```
Decimal
value:              Op-code:           Comment:

  6,   1,           LD   B,  $01       ; set up for VDP register one
 14,   0,           LD   C,  nn        ; load video mode value
205,  32, 253,      CALL $FD20         ; CALL OS write to VDP register
  6,   7,           LD   B,  $07       ; set up for VDP register seven
 14,   0,           LD   C,  nn        ; load background color code
205,  32, 253,      CALL $FD20         ; CALL OS write to VDP register
201                 RET                ; RETurn to BASIC
```

TITLE (asmb#5): Write Data Table To VRAM

```
Decimal
value:              Op-code:           Comment:

 17,   0,   0,      LD   DE, nnnn      ; load VRAM starting address
 33,   0,   0,      LD   HL, nnnn      ; load RAM starting address
  1,   0,   0,      LD   BC, nnnn      ; load byte count
205,  26, 253,      CALL $FD1A         ; CALL OS write to VRAM
201                 RET                ; RETurn to BASIC
```

TITLE (asmb#6): Read Data Table From VRAM

```
Decimal
value:              Op-code:           Comment:

 17,   0,   0,      LD   DE, nnnn      ; load VRAM starting address
 33,   0,   0,      LD   HL, nnnn      ; load RAM starting address
  1,   0,   0,      LD   BC, nnnn      ; load byte count
205,  29, 253,      CALL $FD1D         ; CALL OS read from VRAM
201                 RET                ; RETurn to BASIC
```

```
  10 REM BASIC INIT recover program
  20 REM ONLY USE THIS PROGRAM IF YOU WANT TO
  30 REM ERASE THE CURRENT CATALOG!!!!!!!
  40 LOMEM :28000: TEXT
 100 DATA 62,0,1,0,0,17,0,0,33,0,216,205,243,252,201
 110 FOR x = 27600 TO 27614: READ ml: POKE x,ml: NEXT
 120 PRINT " This program will recover a"
 130 PRINT " freshly INITed datapak or"
 140 PRINT " disk.  Use it with extreme"
 150 PRINT " CARE!!!  It will erase the"
 160 PRINT " current CATALOG!!!"
 170 VTAB 10: PRINT " Which drive?"
 180 PRINT: PRINT " 1 = tape one"
 190 PRINT " 2 = disk one"
 200 PRINT " 3 = abort INIT recover"
 250 GET key$: IF key$ < "1" OR key$ > "2" GOTO 1000
 300 IF key$ = "1" THEN  POKE 27601,8
 310 IF key$ = "2" THEN  POKE 27601,4
 400 VTAB 16: PRINT " really INIT recover "
 410 INPUT " (yes or no)?";yn$
 420 IF LEFT$(yn$,1) <> "y" GOTO 1000
 500 CALL 27600: POKE 27612,246: POKE 27606,1: CALL 27600
 510 PRINT: PRINT " CATALOG recovered!!": END
1000 TEXT: PRINT " program terminated.": END
```

```
  10 REM machine code INIT
  50 LOMEM :28000: TEXT
 100 DATA 62,8,1,1,0,17,255,0,33,223,107,205,189,252,201
 110 FOR x = 27600 TO 27614: READ ml: POKE x,ml: NEXT
 200 INVERSE: PRINT " machine code INIT": NORMAL
 210 PRINT: PRINT " Please select: ": PRINT
 220 PRINT " 1 = INIT medium": PRINT " 2 = abort program"
 230 GET key$: IF key$ < "1" OR key$ > "2" GOTO 230
 240 IF key$ = "1" GOTO 300
 250 TEXT: PRINT " program terminated.": END
 300 PRINT key$: PRINT: PRINT: PRINT " Which device?"
 310 PRINT " 1 = tape one": PRINT " 2 = disk one"
 320 GET key$: IF key$ < "1" OR key$ > "2" GOTO 320
 330 IF key$ = "1" THEN  dv% = 8: vo% = 255
 340 IF key$ = "2" THEN  dv% = 4: vo% = 160
 400 PRINT key$: PRINT: PRINT: PRINT " What directory length (1-3)?"
 410 GET key$: IF key$ < "1" OR key$ > "3" GOTO 410
 420 di% = VAL(key$)
 500 PRINT key$: PRINT: PRINT
 510 PRINT " What name for the volume (any"
 520 INPUT " eleven characters? ";vn$
 530 PRINT: PRINT: PRINT " Really INIT (y OR n)?"
 540 GET key$: IF key$ <> "y" AND key$ <> "Y" GOTO 250
 550 POKE 27601,dv%: POKE 27603,di%: POKE 27606,vo%: PRINT
 560 IF LEN(vn$) > 11 THEN  vn$ = LEFT$(vn$,11)
 570 vn$ = vn$+CHR$(3): lv = LEN(vn$)
 580 FOR x = 1 TO lv: POKE 27614+x,ASC(MID$(vn$,x,1)): NEXT
 600 CALL 27600: cd = PEEK(16821): PRINT: PRINT
 610 POKE 16821,dv%: PRINT CHR$(4);" catalog"
 620 POKE 16821,cd: END
```

```
   10 REM volume name changer
   20 REM does NOT INIT medium
   50 LOMEM :28000: TEXT
  100 DATA 62,0,1,0,0,17,1,0,33,0,212,205,243,252,50,255,107,201
  110 FOR x = 27600 TO 27617: READ ml: POKE x,ml: NEXT
  120 DATA tape one,disk one,exit program
  130 FOR x = 1 TO 3: READ menu$(x): NEXT
  140 start = 54272
  200 VTAB 2: HTAB 2: INVERSE: PRINT " VOLUME NAME CHANGER"
  210 NORMAL: VTAB 4: HTAB 2: PRINT "Which option?": VTAB 6
  220 FOR x = 1 TO 3: PRINT " ";x;" = ";menu$(x): NEXT
  230 GET key$: IF key$ < "1" OR key$ > "2" GOTO 10000
  240 IF key$ = "1" THEN  dv% = 8
  250 IF key$ = "2" THEN  dv% = 4
  260 dv$ = menu$(VAL(key$)): POKE 27601,dv%
  270 VTAB 10: PRINT " Press [RETURN] when ";LEFT$(dv$,4)
  280 PRINT " is ready . . ."
  290 GET key$: IF key$ <> CHR$(13) GOTO 10000
  400 CALL 27600: IF PEEK(27647) <> 128 GOTO 10010
  410 IF PEEK(start+13) <> 85 GOTO 10010
  420 IF PEEK(start+14) <> 170 GOTO 10010
  500 vn$ = "": FOR x = 0 TO 11: pk = PEEK(start+x)
  510 ON pk = 3 GOTO 520: vn$ = vn$+CHR$(pk): NEXT
  520 VTAB 14: PRINT " Volume name is: ";vn$
  600 VTAB 16: PRINT " What new volume name do you "
  610 INPUT " want to use? ";vv$
  620 lev = LEN(vv$): IF lev > 12 THEN  lev = 12
  630 FOR x = 0 TO 11: POKE start+x,32: NEXT
  640 FOR x = 1 TO lev: POKE start+x-1,ASC(MID$(vv$,x,1)): NEXT
  650 POKE 27612,246: CALL 27600
  660 VTAB 20: PRINT " Volume name changed.": END
10000 TEXT: PRINT " program terminated.": END
10010 TEXT: PRINT " CAN NOT ACCESS ";dv$;".": END
```

TITLE (asmb#7): Write Block To Storage Medium

```
Decimal
value:             Op-code:          Comment:

 62,  0            LD   A,  nn       ; load device code
  1,  0,  0,       LD   BC, $0000    ; clear BC to zero
 17,  0,  0,       LD   DE, nnnn     ; load the medium's block number
 33,  0,  0,       LD   HL, nnnn     ; load the RAM buffer address
205, 246, 252,     CALL $FCF6        ; CALL OS write to medium block
201                RET               ; RETurn to BASIC
```

TITLE (asmb#8): Read Block From Storage Medium

```
Decimal
value:             Op-code:          Comment:

 62,  0            LD   A,  nn       ; load device code
  1,  0,  0,       LD   BC, $0000    ; clear BC to zero
 17,  0,  0,       LD   DE, nnnn     ; load the medium's block number
 33,  0,  0,       LD   HL, nnnn     ; load the RAM buffer address
205, 243, 252,     CALL $FCF3        ; CALL OS read from medium block
201                RET               ; RETurn to BASIC
```

TITLE (asmb#9): INIT Storage Medium Directory

```
Decimal
value:             Op-code:          Comment:

 62,  0            LD   A,  nn       ; load device code
  1,  0,  0,       LD   BC, nnnn     ; load the directory block size
 17,  0,  0,       LD   DE, nnnn     ; load the volume block size
 33,  0,  0,       LD   HL, nnnn     ; load start address of volume name
205, 189, 252,     CALL $FCBD        ; CALL OS INIT medium directory
201                RET               ; RETurn to BASIC
```

BASIC loads the volume length from adresses 25305 (low byte) and 23506 (high byte). The default volume length is 255. You may want to change it to 160 when you INIT single-sided disks.

The directory size is at address 25308. The default value is "1". You may want to change this setting to a "2" or a "3".

The volume name, volume length, and directory size are transferred to a data table in the operating system. This table is 104 bytes in length starting at address 62426.

INIT reads block one (the current directory) puts it in a buffer and replaces the first three bytes with a machine code jump to SmartWriter. Then, this buffer is transferred to block zero. Next the buffer is cleared to all zeros. Now the first 104 bytes of the buffer is replaced with the data table. The table contains the four 26-byte system file slots.

You should note that INIT does NOT erase the medium. It only clears the 26-byte slots that describe the stored files.

You can recover a freshly INITed (one block directory) medium by transferring block zero to block one. The files are all still in tact!!!

OS Media Routines:

64758 (246,252) (FCF6)
write block to medium

64755 (243,252) (FCF3)
read block from medium

64701 (189,252) (FCBD)
INIT medium directory

OS 1K Buffers:

54272 (0,212) (D400)
BASIC CATALOG buffer

55296 (0,216) (D800)
BASIC primary file buffer

56320 (0,220) (DC00)
BASIC secondary file buffer

The Directory Programs:

The directory programs illustrate the topics discussed in this issue. The OS buffers itemized above may be used by machine code routines in order to conserve valuable RAM.

The first program on page 18 transfers block zero to a buffer and then transfers the buffer to block one. Thus, the medium is recoverd from INIT. USE THIS PROGRAM WITH GRAVE CAUTION!!

The second program on page 18 shows you how to use the OS INIT routine. However, you may prefer to employ the mentioned POKEs to alter BASIC instead.

The program on page 19 allows you to change the volume name without INITing the medium. If you use the recover from INIT program, you'll probably want to use this one too.

The assembly language lists on page 20 show you the details of setting up the particular OS routines. Have fun . . .

WORD POWER

Did you study last month's words? A refined vocabulary permits clarity of thought. Some psychologists even believe that expanding your vocabulary can increase your intelligence quotient by releasing your dormant abilities. Here are five more.

(1) analogous:

A. related to computers
B. hard to understand
C. very talkative
D. comparable, similar

(2) apprise:

A. a large gift
B. a hybrid apple
C. to inform
D. inducing sleep

(3) detrimental:

A. damaging, injurious
B. loose sediment
C. a different route
D. without conviction

(4) penultimate:

A. the very last
B. second to last
C. to design stationery
D. a large quill

(5) refurbish:

A. garbage, trash
B. to deny
C. to renovate
D. to decline to do

The answers are: (1) D, (2) C, (3) A, (4) B, and (5) C.

PRODUCT REVIEWS

Product:	The Hacker's Guide to ADAM
Manufacturer:	Peter & Ben Hinkle
Media Type(s):	book plus DDP or disk
Graphics rating:	N/A
Instructions:	93
Value for money:	98
Recommendation:	HIGHLY RECOMMENDED
Price:	$17.95
Rated by:	staff

This is a 61-page (full-length) manual for advanced ADAM users. Every page is packed with details about ADAM. Some topics include: the video chip, bank switching, the sound chip, and the OS jump table.

The data pack includes all the programs (18) that are listed in the book. "The Hacker's Guide to ADAM" is an exceptionally good value. If you don't have it, GET IT.

Product:	PowerPRINT
Manufacturer:	STRATEGIC SOFTWARE
Media Type(s):	DDP or disk
Graphics rating:	85
Instructions:	95
Value for money:	85
Recommendation:	recommended
Price:	$24.95
Rated by:	staff

This program is an enhancement to SmartWriter. Features include: variable margins, use of headers and footers, and margin justifications. The files are printed from SmartBASIC and printing is very slow. However, the printed pages look very good.

Product: THE REEDY LIBRARY
Manufacturer: REEDY SOFTWARE
Media Type(s): DDP or disk

Graphics rating:	90
Instructions:	93
Value for money:	85
Recommendation:	recommended
Price:	$24.95
Rated by:	staff

This is a collection of 11 games and/or subroutines. Included is a full length text adventure "Michigana Jones". Some of the subroutines that you can merge with your BASIC programs include: save a GR screen, load a GR screen, and large characters for HGR screens.

Product: ADAM T-shirt decal
Manufacturer: John F. Busby, II
Media Type(s): N/A

Graphics rating:	95
Instructions:	95
Value for money:	95
Recommendation:	highly recommended
Price:	$5.00
Rated by:	staff

This is a very nice looking T-shirt decal featuring the ADAM computer and some of its peripherals. If you're an enthused ADAMite, you'll love this accessory. See this month's bulletin board for ordering information.

BULLETIN BOARD

SOFTWARE FOR THE ADAM
MARATHON COMPUTER PRESS
P.O. Box 68503
Virginia Beach, VA 23455

NIAD ADAM USERS GROUP
P.O. Box 1317
Lisle, IL 60532

ADAM PERIPHERALS AND INTERFACES
EVE ELECTRONIC SYSTEMS
2 Vernon Street, Suite 404
Framingham, MA 01701

ADAM T-SHIRT DECAL
JOHN F. BUSBY, II
6634 SW 41st Street
Davie, FL 33314

DISCOUNT PARTS FOR ADAM
American Design Components
FREE catalog, call TOLL-FREE
1-800-524-0809

SPARE PARTS FOR ADAM
Jameco ELECTRONICS
1355 Shoreway Road
Belmont, CA 94002

The "NIBBLES & BITS BULLETIN BOARD" is free advertising for ADAM supporters. Ads are subject to space availability. Maximum: 45 characters (including spaces, punctuation, etc.) for normal width per line and 22 characters per line for double width. Please limit ads to FOUR lines.

Local ADAM users groups are listed FREE in alphabetical order (by state).

ADAM ACCESS is paid advertising. Write to us for details.

LOCAL USERS GROUPS

FLORIDA:

John F. Busby, II, 6634 SW 41st Street, Davie, FL 33314

Michael G. Graham, 217 Albert Street, Winter Springs, FL 32709

Robert J. Niemeyer, 292 Boca Ciega Point Blvd. North, St. Petersburg, FL 33708

Howard Pines, 812 Pinedale Road, Ft. Walton Beach, FL 32548

GEORGIA:

John Moore, 1970 Fisher Trail NE, Atlanta, GA 30345

HAWAII:

Marc Acosta, 1534 Hoonipo Street, Pearl City, HI 96782

ILLINOIS:

Donald R. Lager, 5415 North 2nd Street, Rockford, IL 61111

KANSAS:

David E. Carmichael, 1325 North Meridian, Apt. 201, Wichita, KS 67203

Joe Reardon, 1513 Tauromee, Kansas City, KS 66102

KENTUCKY:

Keith Bowman, P.O. Box 434, Alexandria, KY 41001

Intel-BEST 3.3 by DIGITAL EXPRESS, INC.

In the fall of 1985 DATA DOCTOR developed SmartBEST V1.0. SmartBEST adds 27 enhancements to BASIC: sound, graphics, etc. SmartBEST was DATA DOCTOR's fastest selling software. It is a great program!

Having purchased the copyrights to all DATA DOCTOR software, DIGITAL EXPRESS, INC. has improved the program dramatically. Intel-BEST 3.3 makes over three dozen changes to SmartBASIC.

Nearly all BASICs permit the question mark as a shortcut for the PRINT command. Intel-BEST 3.3 includes five similar shortcuts: 'F' for FLASH, 'H' for HOME, 'I' for INVERSE, 'N' for NORMAL, and 'T' for TEXT. With these added shortcuts, it's a lot easier and faster to enter BASIC programs.

Intel-BEST even corrects the DATA and REM spacebump bug!!! It also corrects the COLOR, HCOLOR, and SCRN color tables so that you only have to use ONE color code chart. Intel-BEST permits you to enter up to 216 characters per program line (BASIC limits input to 120). Intel-BEST eliminates many of the unnecessary spaces added with the LIST command. And Intel-BEST resets the POKE limit to 65535.

Intel-BEST also includes several ready-to-use machine code routines. Included are a block read and a block write routine. Also included is a routine that will read the catalog (block one) without interrupting a RUNning program. Routines are also included that will instantly change the background color and the NORMAL and INVERSE colors without clearing the screen. Intel-BEST even includes a routine that will instantly change the graphics window color in GR or HGR mode without clearing the screen.

Intel-BEST includes a relative RESTORE command (LINE) that will let you RESTORE to any line number that you specify. Using this command gives you powerful control over your DATA. Also included is a new command (Ln8) that will increase the TEXT window in either graphics mode to eight lines instead of four.

Intel-BEST 3.3 also includes some powerful audio enhancements. ADAM will permit four simultaneous sounds (three voices and a noise). Intel-BEST makes it very easy for you to get the maximum benefit from ADAM's sound capabilities. With each voice you can use any of 1024 possible sounds. The noise generator is equipped with 8 built-in sound effects. Of the 1024 sounds possible with each voice only 48 correspond exactly to musical notes. Intel-BEST provides you with direct access to these musical notes so that you can easily enter music from songsheets. The extensive instruction manual fully explains how to enter these notes and how to create special noises and sound effects. To enter a 'C' note in the lowest octave (of the four standard octaves) in the first voice all you need to do is: T1 = 3 [RETURN]. Changing the volume is equally simple; just enter the volume setting (0 through 15). Intel-BEST includes nine easy-to-use music commands (V1=, V2=, V3=, T1=, T2=, T3=, NV=, NS=, and OFF).

Intel-BEST 3.3 is the most elaborate enhancement to SmartBASIC ever developed for ADAM and it's also the least expensive. Intel-BEST will also RUN just about ALL SmartBASIC programs. Once you've tried Intel-BEST 3.3, you'll NEVER want to go back to SmartBASIC V1.0. Intel-BEST is a pure machine code program (1742 bytes) and it sets LOMEM to 27600.

The Intel-BEST 3.3 software medium includes 10 demonstation programs. These include a nice computer version of 'Jingle Bells' and a very useful media copy utility.

DEI PRODUCT LIST:

Software:

Intel-BEST 3.3 (dynamic enhancement to SmartBASIC V1.0)
$18.95 -- datapak only

Intel-LOAD (converts BASIC programs to LOAD up to 12 times faster)
$11.95 -- datapak only

HARDWARE:

DEI blank datapaks (DEI's datapaks are as reliable as Coleco's datapaks)
$2.75 (each)
$24.95 (for ten)

ACCESSORIES:

adhesive labels (tractor-feed, fan-fold, 3 1/2 x 15/16, single-column)
$2.25 (500 labels)
$3.95 (1000 labels)

blank white paper (tractor-feed, fan-fold, 9 1/2 x 11, 20# wt., 250 sheets)
$5.95

DEI EZ-REFERENCE GUIDES:

#101 (appr. 700 numeric Z-80 instructions; decimal/hex/op-codes/operands)
$1.00 (with business size SASE, no shipping)

#102 (appr. 700 alphabetic Z-80 instructions; decimal/hex/op-codes/operands)
$1.00 (with business size SASE, no shipping)

NIBBLES & BITS SUBSCRIPTIONS:

$18.00 (one year -- 12 issues)
$12.00 (six months -- 6 issues)

SPECIAL NOTICE: All DEI datapaks are warrantied to be free from defects in material and workmanship. If the storage medium proves defective, return it to DEI for a replacement. This warranty applies to datapaks purchased 'blank' and to those purchased with DEI program(s) stored on them. DEI shall, under no circumstances, be liable for consequential damages arising from use or misuse.

NOTE: The prices listed above are effective 8-1-86 thru 8-31-86.

PRODUCT ORDER FORM:

YOUR NAME: ______________________________________

ADDRESS: ____

CITY: ____________________ STATE: __ ZIP: ________

PHONE NUMBER: ___________________________________

SUBSCRIPTION ID NUMBER: __________________________

< ITEM/QUANTITY/MEDIA >	< PRICE >
<____________________________>	$<___.__>
<____________________________>	$<___.__>
<____________________________>	$<___.__>
<____________________________>	$<___.__>
<____________________________>	$<___.__>

SUBTOTAL:	$_____.__	
SHIPPING:	2.50	
TAX:	_____.__	(NC residents only -- 4.5%)
OTHER:	_____.__	
OTHER:	_____.__	(subscription/renewal)
TOTAL:	$_____.__	

To order: complete this form, and send check or money order (US FUNDS) to:

DIGITAL EXPRESS, INC.
1203 Northwoods Drive
Kings Mtn., NC 28086

Coming soon . . .
ADAM printer ribbons, disks for ADAM, Coleco ADAM software, 3rd-party ADAM software, and more!!!

HACKER'S CONTEST

The NIBBLES & BITS Hacker's Contest is a monthly competition. The winner of each contest is randomly selected from the correct responses postmarked within the specified dates. No individual shall be named the winner in three consecutive contests. The winner of each contest shall be awarded ten dollars and a free three month extension to his/her NIBBLES & BITS subscription term. Decisions of the judges are final.

Responses for this month's contest will be considered valid if, and only if, they are postmarked after July 31, 1986 and prior to September 1, 1986. The winner shall be announced in the October issue of NIBBLES & BITS.

Write a SmartBASIC program (it may include machine code in DATA statements), which will save an HGR screen on data pack or disk. Good luck . . .

SOFTWARE EXCHANGE

Our first two public domain libraries are completed. Each BASIC PD library contains over 70K of programs. Each library includes two INSTRUCTion files which can be read or printed from SmartWriter. All BASIC program are speed-RUN. Most of the programs are controlled from a "ramdisk" written primarily in machine code.

Each library is available on datapak for $7.95.

To get a free copy of a specific library: (1) contribute an original program, (2) send a signed statement that the program is not copyrighted, (3) send the program on a data pack, (4) request the specific library that you want in return. "FAMILY COMPUTING" programs are not accepted.

These first two volumes are: B-1.0 and B-2.0. We will list the contents of one volume per month beginning with the September issue.

SWIFT POLL BALLOT

As a NIBBLES & BITS subscriber, you are invited to submit one SWIFT POLL ballot per month. This ballot must be cut out (not duplicated) and postmarked prior to Sept. 1, 1986 in order to be verified as valid. The results for the current poll shall be tallied and made public in the October issue of NIBBLES & BITS. Please list your top ten software preferences for the month of AUGust, 1986 in the order that you like them (best first, next best second, etc.).

YOUR NAME:________________________ SUBSCRIPTION ID NUMBER:________________________

1. ______________________	2. ______________________
3. ______________________	4. ______________________
5. ______________________	6. ______________________
7. ______________________	8. ______________________
9. ______________________	10. ______________________