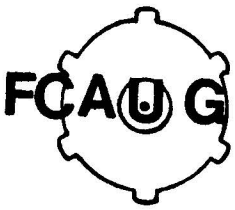
SYNTAX 1.3

JULY – AUG. 1985

TABLE OF CONTENTS



FIRST CANADIAN ADAM USERS' GROUP
P.O. Box 547 Victoria Station
Westmount, P. Q.
H3Z 2Y6

E D I T O R I A L

For those of you that thought the game was over when Coleco dropped the Adam, think again. People who know the computer industry expect lines to drop. They also know that this is often the best thing that can happen to the users. Third party manufacturers have and always will begin to creep in with "I-didn't-think-it-could-be-done!" product about 6 months after a line is dropped. So why should Adam be any different?

In fact, it's not. We are on the verge of the greatest outpouring of Adam compatible product ever. And it is because Adam was dropped that this is so. Why? Simply because third party developers now feel secure enough to develop their own product. They no longer have to worry about being beaten to the punch by a giant conglomerate with hundreds of researchers at their beck and call.

In this issue we look at the first such third party peripheral, developed in Canada: the 64K memory expander. There are many more products on the way in Canada and in the States. As we get to try these out we will report them to you. As of now we know that RS-232 adapters have been developed. There are also tractor feeds, 80 column cards, and lots of programs.

Why is Adam product from Coleco so hard to find? Although third party importance is steadily growing greater Coleco remains and will for a little while yet, the source of software and peripheral for Adam. So the above question is still very much an important one. Apparently, Adam product is disappearing from stores all across Canada. Is it the stores or Coleco that is responsible for this turn of events? We beleive that it is the latter. It seems that Coleco is recalling all its product as of now. They will then sell their inventory in Canada to Canadian Tire at the latest, in October of this year. How do we know this? In attempting to buy product in bulk from Coleco for the club, we were told, "No deal, it's all going to Canadian Tire". Looks like we'll have to wait till then to find the stuff we want in our neighbourhood stores. Till then, however you can order, by mail, anything you want, directly from the company. Write to Coleco Canada, at 3700 St Patrick St in Montreal (H4E 1A2) for a catalog. Incidentally, you can also get Adamlink 2 in Canada now, for $15.00 Canadian at the same address.

Next issue, we intend to publish a complete listing of all local Canadian users' groups for the Adam. So send in your information from your area as soon as possible. Don't forget to send in a name so that we can also verify the group's existence.

Rumor has it that there is or will be a bulletin board opening up in Toronto and Montreal. The Montreal one will be national.

People who wish to contribute programs to FCAUG. Please, if at all possible, send your programs in on disk or tape. We get quite a few and try to verify them all. If you send us up and running

programs not only do we not have to retype them, but it also speeds up the verification process considerably. In any event, we would like to see more programs sent in so that we can build up a good program library. Don't worry that your programs aren't good enough or are of limited interest. We can all learn from each other.

Our first package of users' programs will be ready by next issue for a price of $10.00. This includes the tape or disk plus the mailing costs. A list of the programs will appear next issue. If you don't want to wait you can order now, but we recommend waiting for the complete listing.

Some people have had problems getting their SYNTAX 1.2 on time. Part of this is due to us getting a little behind schedule. However, we are finding that some issues are not getting to their intended destination quickly enough. In some cases, the post office has sent them back to us instead of forwarding them to the subscriber. We believe that this is related to our attempt to get second class mailing privileges. We have taken steps to have the post office remedy the situation. We apoligize to those members that have experienced delays in getting the second issue. If you have yet to receive your copy write to J.D. Moore. He'll put one in the mail to you right away.

A reminder... Since introduced in our first issue of SYNTAX back in March, many members have taken advantage of our software back up service. We have provided you with extra copies, or disk versions of SmartBasic, AdamCalc, personal data bases, etc. We can and will back up any of your Adam software (excluding super games). Send in the software package plus the tape or disk that you want it backed up onto along with $10.00. The procedure remains unchanged (see p. 6 in SYNTAX 1.1) except for the price reduction. If you can't part with your program, send in a photo-copy of your bill as proof that you have actually bought the item and we will copy it as per your instructions. If you want FCAUG to provide the blank tape or disk, send $15.00. We offer this service to our members but remind them that there are now software-copying programs that can be purchased from the States for anywhere from $50.00 to $75.00 U.S. We provide you with an alternative. The choice is entirely yours.

This issue was a collective effort by: J.D. Moore, Ron Saunders, Peter Kopystecki, and Andrew Wiles.

- - -

SYNTAX is published bimonthly by First Canadian Adam Users' Group (FCAUG), P.O. Box 547, Victoria Station, Westmount, Que., H3Z 2Y6. Subscriptions: 6 issues for $20.00. Second class mail registration pending. POSTMASTER: Send address changes and notice of undelivered copies to above address. Return Postage guaranteed.

As an item of interest to all Adam users

If you are having trouble obtaining the Accessories Kit with the head cleaner, I have found that the Radio Shack Head Demagnitizer and Cleaner Kit; part # 44-8016 does a very passable job as long as holes are drilled into the cassette to match the locating holes in the Data-settes used in the Adam data-drive.

I found that after about 20 hours of use the computer refused to load programs due to dirty heads and the Colecovision cleaner is very hard to find in the Maritimes, so I looked for an alternative solution and found a reasonably priced substitute.

I hope this may help some Adam owners in the near future. The price of the cleaner is only $5.95 and a quarter hours' worth of work as compared to the $60 cost of the Coleco kit.

Karl Korber

Although your main subject is head cleaners, you bring up other things worthy of further comment. And strangely enough these subjects are all related. First let me talk about head cleaners. There are many types of head cleaning kits available and they all do as good a job as the next. You can use head cleaning kits designed for audio tapes or even other computers. As long as you are using a liquid head cleaner, this rule holds true. Most liquids used are alcohol or things like dichloro-dibromo ethane. The chemical in Adam's head cleaning kit is isopropyl alcohol (rubbing alcohol). I have seen other alcohols used (usually the sole ingredient). On occasion, I have even used ethyl alcohol (ie liquor: a clear spirit like vodka is good because it has few additives and is easily available, better yet is Alcool or another very pure ethyl alcohol.) If you want, you can buy the chemical used in the Adam head cleaning kit in bulk. Usually you can get it in a jug at a very low price. (Incidently it has just occurred to me that people who know nothing about chemistry can get themselves into a bit of trouble. So: although the aforementioned chemicals are alcohols and chemically similar, etc. they are only equivalent in the sense that they can clean ferric oxides off magnetic tape recorder heads. Don't drink them. If you do don't expect to be around long enough to tell anyone about the experience.)

Use a Q-tip to apply the liquid (not too much). Since these liquids have a low boiling point (somewhat volatile) they evaporate very quickly. That means that you don't have to wait very long till they dry but also that several applications may be necessary. So apply and wipe until that head sparkles!

Point number 2. Some head cleaners come dry on a cassette. Although more convenient, they are often not as good as liquid cleaners. The tape and cleaning solution wear themselves out and eventually your heads don't get clean at all. Nevertheless the real point here is that if you so desire this type of cassette-cleaner can be used on Adam by drilling holes in the back of the shell in order to match the holes found in Adam's shell. (From your letter I can see that this is exactly what you've done.) If we extrapolate further, we see that if this can be done for a head cleaning shell there's no reason that it can't be done for an audio shell. Purpose? To make your own digital data packs. The only problem now is how to format the tapes. There are some copying programs available in the States now, so if you copy a blank tape to a modified audio tape it should work. Martin Consulting sends out their programs on a modified TDK tape, so it is possible. (As another aside we are looking into either convincing the developer of our software copying program to market his or arrange for Canadian distribution of one of the American programs).

The last point is tape head demagnitization. Every school kid knows that if you rub a magnet against a piece of steel long enough you will magnetize the metal. Well this is exactly what a magnetic tape is doing to your tape heads. Keep using your tapes without demagnitizing them and very soon you will very efficiently erase all of your valuable data. So occasionally demagnitize your tape heads. The procedure is simple and since it's the same as it is for audio, I suggest you consult a stereo store or magazine for the specific steps.

Some of you more astute readers may recall that I said that the points brought up in this letter were all related. And they are. What I've found is that all too often simple things are mystified in order to confuse the customer. Companies have been known to force their customers to either pay for something that is not worth a fraction of what it costs or they try to make the customer totally dependent on the company such that he/she thinks that the product is so specialized that the parent company is the only one that can supply it. I think that this is what we Adam users have been victim of. The head cleaning kit and tapes (for they are simply just tapes and not the mysterious sounding "digital data packs") are just two examples of this outrage. We'll only be too happy to expose any more similar examples in this and upcoming issues. (Later in this issue, see the article on regenerating cartridge ribbons for the printer.)

Dear FCAUG

I wonder if I could get a program to figure out the daily interest on a thousand dollar certificate at 13.25% interest. I have been unsuccessful to date to make my own program and unable to find anyone to help me.

T.W. Roodway

Reviewing my economic notes, I came across the following equation for compound interest: F=P(1+i)^n where:
F=future amount
P=principal
i=interest
n=# of time periods
The following program incorporates the equation and prints out the day, interest, and cumulative principal. I hope this solves your problem.

```
5 DIM f(365)
10 READ p,d
15 f(0)=p
20 FOR n=1 TO d
30 f(n)=p*(1+.1325/d)^n
40 int=f(n)-f(n-1)
50 PRINT n, f(n), int
60 NEXT n
70 DATA 1000,365
```

Dear Sirs:

I recently purchased a copy of "Programming Adam" by Edward B. Chaflin and John S. Heil: published by Banbury Books Inc., Wayne, PA.
I found the book well written, highly instuctional and refreshing after the manual supplied with Adam.
I programmed in the MIXNMATCH game which seemed like an interesting challenge for a novice programmer. I was quite frustrated when having thoroughly debugged the program and compared a printout to the book; I still couldn't get it to work I then called up my son who looked at it for a few minutes, said "This program can't work with line 500 as it is, it isn't logical." He then proceeded to rewrite line 500 and added line 495. We now had a running program which also accepted either upper or lower case input. He also changed line 90 to make the word selection more truly random.
He's the scientist and I'm the artist so I then got busy centering lines on the screen eliminating words which could be spelled to make more than one word and I then added colour to some screens thanks to the March issue of SYNTAX.
There was a vast number of line changes in order to make the game more attractive and allow for the absence of line 1 on most TV sets.
If you feel that you can print the entire revised program (with due credit to the authors of course) I'm sure that many members would enjoy it. If not perhaps you could publish the revised lines which I have marked. It would be a help to many of the members who own this book.
The following numbered lines are the ones I have modified, added or deleted. The ones marked with an * are vital to the program, the others were cosmetic or word replacements.

10	90*	150	250	495*
24	104	160	252	500*
25	105	170	324	660
50	120	180	325	670
70	130	210	340	680
80	140	211	350	

Your first issue of SYNTAX was excellent, hope you can keep up the standard.

J.T. Addison
Scarborough, ONT.

See the programming section for the Addisons' modified MIXNMATCH program.

BOOK REVIEWS: by Kel Schwab

Coleco Adam users handbook :
Weber Systems Staff
Balantine Books 1984 $12.95

Not a bad book for an all around introduction to the computer. Well written and full of ideas, it takes the novice through all the steps necessary to get started in computing. It covers setting the system up, word processing and writing programs in basic. It also contains listings and documentation for four useful programs. These are: a metric converter, loans analysis, break even analysis and an electronic phonebook. Details on the file handling routines are not clear and can lead to problems. A good beginner's book and better written than the manuals supplied by Coleco. In short, easy to read with useful programs.

Adam's Companion:
by R. Benson & J. Rochester
Avon Books 1984 $9.95

This is the best book that I have read to date that covers the complete ADAM system, from set-up to advanced programming in basic. Clearly written text and well documented programs make this one of the most used books in my computer library. The programs in this book include, a low resolution graphics picture editor, a simple video game, a music-maker program and a mailing label system. The technical details of the machine and all the basic commands are clearly explained. This is one good book!

(editor's note: Although these 2 books are very good, they are not without their faults. We'll look at these books and others in upcoming issues of FCAUG.)

Adam Sound Programming

Last issue, we talked about how to produce sound on the Adam. You are now able to produce a single note of any duration, frequency and loudness you desire. If you feel like you're ready to delve a little further into the murky waters of sound programming, read on. In this article, I will explain how and why sound programs work. Once that's done we'll put all the single notes that we've created into something called "music". And finally we'll look at how to produce different sound effects using the "noise channel". That should clear up the water a bit.

First thing to do. Get issue 1.2 and familiarise yourself with the sound program. This is the ultra basic model. Everything else is a variation on this theme. Remember this and you won't ever get confused. (The present form of the program will give you an "out of data" error message when you try and get a second note. To fix this add the line "restore" just before the last line "goto 20". This will make the read statement in line 60 start from the first data statement instead of the one after the last, as it is doing now.)

Finished? Good. I hope you noticed how easy this program is once you break it down into discrete modules. After you set the lomem, there are 3 sections. First you input the note variables. Next you convert these values into a form that the "assembler" section will understand. Finally the assembler subsection is CALLed into action. It takes the converted values and sends them to the sound chip (TI SN76489A). Sound is produced after all the parameters are sent to the sound chip. Control is then passed back to the Z-80.

The sound or music assembler portion of a sound program never changes. It is the pipeline that sends information from the Z-80 to the sound chip. You can set it up before the main sound program because it is not used until it is CALLed into use. Another analogy: think of yourself as being on one mountain top and trying to send something to another. Well before you can start you have to build a bridge. Then when you want to go or send something to the other mountain top you simply send it over the pre-existing bridge. This is all the assembler section is - the bridge between the CPU and the sound chip.

Let's take a closer look at this assembler subsection. Understanding something about assembly language helps but this routine is so small it should be comprehensible to all of you. Register A (a temporary loading area within the Z-80) is loaded with the memory contents found at memory location 28006. This value is then shipped out to the sound chip. Where the CPU is designed to output to other registers and memory locations, the sound chip is designed to output to a speaker. So assuming you have sent the proper numbers the sound chip will activate the speaker. That's all there is to this section.

The next section, the main sound program, is the most interesting. This is where your inputted sound parameters are modified in order that they may be understood when sent to the

sound chip. Its too bad that the sound chip does not accept the pitch value as you inputted it. There are 2 reasons for this. The first involves "the internal clock". A new pitch is recalculated in line 110 by multipying your inputted pitch by 32 and then dividing the sum into the frequency of the internal clock signal (3,597,000 cps). This new pitch value is used to set the frequency of the note. It is placed in one of the registers of the sound chip.

The sound chip sends out a frequency by starting at the new pitch value (found in the register of the sound chip) and counting backwards. When it hits zero a vibration is sent out. The count then restarts. The lower the new pitch value, the more times the count reaches zero per unit time, therefore the higher the frequency.

So far so good but where is this new pitch value used in the program? It is used to calculate two more variables "first" and "second". It is these new variables that are then used to define the pitch to the sound chip. This is the second modification on the original pitch input and is a consequence of the internal architecture of the Z-80 CPU used in the Adam. Briefly, the problem is this. The Z-80 is an 8 bit microprocessor. Eight bits are used to represent any one number. The number of permutations of 0's and 1's that can be represented using only 8 positions is limited. This means that the largest number that can be represented is 255, assuming we start at zero (which we do). Why only 8 positions? Convention. These 8 positions are bits, 8 bits equal one byte. One byte is contained at one address location. Therefore if you want to represent a number larger than 255 (which we do) 2 bytes are required (hence "first" and "second"). When these 2 variables are sent to the sound chip the new pitch number is reconstructed in one of the registers there. So the variables "first" and "second" are calculated for the single reason of getting the new pitch value over to the sound chip within the confines of an 8 bit environment. If you progress further within the assembly language jungle this splitting of large numbers will become second nature. Just remember this is not part of the sound program per se but done on behalf of the environmental restrictions of using an 8 bit microprocessor like the Z-80 in the first place.

THE HEART OF THE MATTER

The sound chip has 3 sound channels and one noise channel. So far we have only used the first sound channel. Each channel has a frequency and volume control. Let's look at our original program again in order to see how the sound channel that we used (sound channel one) was activated.

When we created the new variables "first" and "second" we split the new pitch value in a very specific way. You already know the reason for this now here's the how. "Second" is created by dividing the new pitch value by 16. This is done to "shift out" the right most numbers of the new pitch value when it is

represented as a hexadecimal number at the computer level. (Much like dividing 1000 by 100 would give you 10, the 2 right most zeros are "shifted out".) "First" is a little more complicated. It also involves choosing what sound channel we will be using, as well as telling the computer that its number is the right most part of the number when the new value number is reconstructed in the sound chip, as well as carrying that number in the final variable "first". This is the line in question:

```
130 first = 128 + (pitch-second * 16).
```

The "128" sets the leftmost bit to 1. This signifies that this byte will be looked at as either the first or right hand side of the number (low order byte), which it is or the only byte (which in this case it most certainly is not).
To pick a sound channel (one of three) we have to add this number to the value of "first". Since the value we add is zero it is not listed in line 130. I will give you a new line 130 very soon. The last part, if you look at it closely, is nonsensical. This value will always be zero. The program will work but instead of giving you 440 cps (if that's what you asked for) you will get a note a little lower. This is a mistake in the original program and leads to inaccurate frequencies. (Although a frequency will be heard!) Here's the fix. Instead of multiplying the numeric variable "second" by 16 in line 130, multiply the integer variable "second" by 16. This will give you the correct right most bits. The old way gives you the right most bits, but it gives them to you as zeros. Here is the new line 130. It incorporates the change just mentioned and the change to allow you to address the other 2 sound channels.

```
130 first = 128 + (pitch- (int(second)*16)) + register
```

To activate the frequency in channel one define the register variable as zero. (Remember that the value for an undefined variable is zero, therefore if you add line 130 and do want to access channel one it will work without your defining "register". To access the other channels you must find a way to add the proper values to access the desired sound channels.) To access channel two, enter 32. Channel 3, 64. That takes care of frequencies.

Lets look at volume. Our ultra basic model has "hard wired" full volume on channel one as soon as the frequency is sent to the sound chip (line 160). After the delay, zero volume is set to channel one. This means that channel one is still producing its frequency but that we can't hear it (line 180 sets volume to zero in sound channel one). How are these numbers for full and zero volume in channel one picked? Here is the formula:

```
Volume level = 128 + register + decimal value.
```

The 128 we know already. It tells the sound chip that just one byte will be sent. The "register" is similiar to "register" for frequency but it obviously controls volume. To access channel one

it is set at 16. Channel two is set at 48 and channel three at 80. The decimal level is set at zero for full and at 15 for no volume at all. The integers in between these two extremes produce a discrete gradient of decibel levels. Thus full sound, channel one equals 144 because 128 + 16 + 0 = 144. Same channel, no sound = 128 + 16 + 15, ie 159. And that takes care of volume. Duration is, as you have already seen, controlled by the delay loop (line 170). That's all there is to it. You now understand sound programming on your Adam. Following this article you will find some example sound programs to make it all crystal clear to you. Next, we'll discuss one of my favorite topics.

NOISE

Noise is an interesting topic. It is used in many video games and this channel, used in conjunction with the sound channels can give you some very beautiful sounds. The noise channel is activated in a way similiar to the procedure that we've just learned.
Let's look at how to set the volume in the noise channel. What do you know? It's the same formula as it is for sound. When you get to "register" enter 112. The procedure for picking the types of noise that you want is similiar to picking the frequency you want for a particular sound channel. However only one byte is used, so now you know that you have to add 128 to your total right away. You now have to choose the noise channel,ie "register", so you add another 96. You can then choose between continuous (add 4) and periodic noise (add 0). Next you can decide on a "noise shift rate". To try these on the noise channel add 0 to get a nsr of n/512, 1 for a nsr of n/1024, 2 for a nsr of n/2048, and 3 to vary the noise rate with the voice 3 frequency, where n = 4,000,000 cps; Adam's clock rate. I'll give you a few charts to make all this information a little bit easier to apply. Then I'll provide a few exemplary programs. After that you're on your own. For sound programs, see pp. 19 and 20 of this issue's program section.

fig.1 Register - Register Value Chart

Register	Register value
sound channel #1 (frequency)	0
sound channel #1 (volume)	16
sound channel #2 (frequency)	32
sound channel #2 (volume)	48
sound channel #3 (frequency)	64
sound channel #3 (volume)	80
noise channel	96
noise channel (volume)	112

Adam Graphics : Sprites

When you look at the space ships, missiles, and cartoon-like characters in one of Adam's Super games, what you're most probably looking at are sprites. Sprites are graphic entities

that are built into the Video Display Processor (VDP). As you remember, the VDP is the processor chip (TI's 9918) which takes care of all of Adam's graphics.

The VDP is capable of displaying up to 32 sprites (sprites 0 to 31). Each sprite is given its own plane by the VDP. Sprite 0 gets the plane that's farthest in the foreground while sprite 31 is placed that many planes behind sprite 0's. In addition, there are 2 other planes behind the 32 sprite planes. The 33rd plane is the Pattern or Multicolor plane. This is the plane that SmartBASIC uses for all its text, low resolution, and high resolution graphics. The plane behind the Pattern plane is the Border or Backdrop plane and is always of one color.

The sprite screen is divided into a 256 column by 192 row grid. You position a sprite on the screen by giving it a row (y) value and a column (x) value. Each sprite has a name or code associated with it. Each name value corresponds to a particular 8 by 8 dot pattern, thereby giving the sprite a shape. The patterns are stored in the Sprite Pattern Table which works just like the Character Pattern Table (see previous Adam Graphics article). A sprite can be one of 16 colors (one of the colors being transparent). These four attributes make up a sprite definition. They are contained in a 4 byte sprite attribute list in the following order:

Y location, X location, Pattern name, Color

Each 4 byte attribute list is placed in the Sprite Attribute Table along with the (up to) 31 other 4 byte sprite attribute lists. Sprite 0's attribute list is first in the table while sprite 31's is last.

If a sprite's x value is zero the sprite will be on the extreme left of the screen, if it is 255 then it will be on the extreme right. If its y value is zero the sprite will be at the top of the screen while 191 will put it at the bottom of the screen. A sprite can have a y value from 192 to 255 but it will not show up on the screen. This is useful when you want a sprite temporarily off the screen. The Sprite Attribute table is ended by placing a value of 208 decimal or D0 hex into the y attribute byte of the first sprite you don't want to use. This sprite and all other sprites which would have followed in the Sprite Attribute Table will not be availible for use. This conveniently reduces the number of availible sprites to a value less than 32.

Sprites can come in 4 sizes. A normal sprite is like a normal text character, 8 by 8 dots. A double sized sprite is 16 by 16 dots. This means it needs 32 pattern bytes to define it. Both of the previous 2 sizes can be magnified to twice their normal size. This is done by making each dot twice its normal size (ie. magnified dot = 2 by 2 normal sized dots).

Inside the VDP are 9 special locations. Each location is one byte long. The locations are referred to as registers 0 to 8. Note that the registers are inside the VDP chip and not in the VDP's RAM (VRAM). This allows the VDP circuitry to quickly access the

registers. The registers control such things as the location of the various tables in VRAM, the border color, and the size of sprites. To specify the size of all sprites you put the following numbers into register 1:

224-gives normal 8 by 8 dot sprites
225-gives magnified normal sprites
226-gives double sized 16 by 16 dot sprites
227-gives magnified double sized sprites

Register 8 is the status register, it contains a flag bit that goes on whenever 2 sprites come into contact on (or off) the screen. It is known as the collision flag. These are the only 2 registers we will concern ourselves with for now.

The advantage of sprites over the graphic characters created by SmartBASIC's hi res graphic shapes is that of speed of movement and ease of definition. Sprites are part of the VDP's hardware while the hi res shapes are part of the software of the BASIC interpreter. Since hardware is inherently faster than software, sprites can be moved faster and without flicker on the screen. The defining of hi res graphic shapes is unnaturally difficult, it relies on using some ridiculous vector codes. Sprite shapes are defined like normal text characters and result in a simpler pattern definition.

Now comes the bad news, because as nice as sprites are they have no direct support in SmartBASIC. But all is not lost since you can create machine code routines to access them. Following this article is a BASIC program that contains 4 machine code routines which can be used to access sprites. The machine code routines are placed in memory starting at 28000, one routine follows another. The routines make use of a 32 byte buffer in RAM. This buffer starts at 28046 and ends at 28077. The LOMEM : 28100 allows for the routines and buffer to co-exist with BASIC. The VREAD routine will read from 1 to 32 bytes from VRAM into the buffer in RAM. The VWRITE routine will write from 1 to 32 bytes from the buffer to VRAM. The number of bytes to move for both these routines is poked into location 28016. The video address involved is poked into location 28013 (low byte of video address) and location 28014 (high byte of video address). After poking the appropriate locations you access the VREAD routine by the instruction CALL 28000 and the VWRITE routine with CALL 28004. The COINC routine will test when 2 or more sprites are touching each other on the screen. If they collide the byte at location 28045 is set to 1 (true) else it will be reset to zero (false). To test for sprite collision do the following, CALL 28022:IF PEEK(28045) THEN a collision. The WRITEVREG routine places a data byte in a VDP register. The register number is from 0 to 7 (register 8 or the status register can only be read) and is poked in at location 28040. The data byte is poked in at location 28039.
The BASIC program will create 2 sprites (lines 430-600) on the same line and at opposite ends of the screen. Both sprites share the same pattern name . Lines 610-680 move one sprite left and the other right. This is done by either decrementing or incrementing the x attribute. Lines 700-710 check for sprite

collisions and will either magnify or demagnify the sprites when one occurs.

```
  100 LOMEM :28100: TEXT
  110 vr% = 28000: vw% = 28004: coinc% = 28022: wvr% = 28038
  120 bcount% = 28016: buf% = 28046: vaddr% = 28013: vreg% = 28039: cflag% = 280
45
  130 &      load machine routines
  140 FOR i = 0 TO 44: READ dat%: POKE 28000+i, dat%: NEXT
  142 &
  144 &    assembly code of video access routines
  150 &       VREAD 28000  LD A,1Dh
  160 &                    JR 2          goto J0
  170 &      VWRITE 28004 LD A,1A
  180 &      J0           LD (6D73h),A
  190 &                   LD HL,6D8Eh   HL=buffer address
  201 &    VREAD & VWRITE
  210 &                   LD BC,1       BC=byte count
  220 &                    CALL FD00h    read or write to VRAM
  230 &                   RET           return to BASIC
  240 &      COINC 28022  LD HL,6D8Dh   HL=collision flag address
  250 &                    CALL FD23h    put VDP register 8 into A
  260 &                    BIT 5,A       test bit 5 of A
  270 &                   JR NZ,3       if bit<>0 then J1
  280 &                   LD (HL),0     else collision flag=0
  290 &                   RET
  300 &      J1           LD (HL),1     collision flag=1
  310 &                   RET
  320 &      WVREG 28038  LD BC,0       B=video reg. #,C=data
  330 &                    CALL FD20h    put data in VDP reg.
  340 &                   RET
  350 &      COLLISION FLAG = 28045
  360 &      BUFFER = 28046-28077
  370 &
  400 DATA   62,29,56,2,62,26,50,115,109,33,142,109,17,0,0,1,1,0,205,0,253,201:
&   VREAD & VWRITE
  410 DATA    33,141,109,205,35,253,203,111,32,3,54,0,201,54,1,201: &   COINC
  420 DATA    1,0,0,205,32,253,201: &    WVREG
  430 &    give standard sprites
  440 POKE vreg%, 224: &    data byte
  450 POKE vreg%+1, 1: &    write to register 1
  460 CALL wvr%
  470 &    write 8 byte sprite pattern to Sprite Pattern Table
  480 FOR i = 0 TO 7: READ dat%: POKE buf%+i, dat%: NEXT
  490 DATA     60,126,255,255,255,255,126,60
  500 POKE bcount%, 8: &     write 8 bytes
  510 ad% = PEEK(64870): POKE vaddr%, ad%: &      write low byte of       Sprite
 Pattern Table
  520 ad% = PEEK(64871): POKE vaddr%+1, ad%: &     write hi byte of  SPT
  530 CALL vw%: &    write 8 bytes from buffer to VRAM
  540 &    write 2 sprite attributes to SAT
  550 FOR i = 0 TO 8: READ dat%: POKE buf%+i, dat%: NEXT
  560 DATA    90,1,0,7,90,200,0,12,208
  570 POKE bcount%, 9: &    write 9 bytes
  580 lsat% = PEEK(64868): POKE vaddr%, lsat%: &     low byte of          addres
s
  590 hsat% = PEEK(64869): POKE vaddr%+1, hsat%: &     hi byte of          addres
s
  600 CALL vw%: &    write 9 bytes to VRAM
  610 &    move sprites accross screen
  620 x0% = 1: x1% = 200: POKE bcount%, 1: POKE vaddr%+1, hsat%: spsize% = 224:
d = 0
  630 x0% = x0%+1: IF x0% < 256 THEN 650
  640 x0% = 0
  650 x1% = x1%-1: IF x1% > -1 THEN 670
  660 x1% = 255
  670 POKE buf%, x0%: POKE vaddr%, lsat%+1: CALL vw%
  680 POKE buf%, x1%: POKE vaddr%, lsat+5: CALL vw%
  684 IF d = 0 THEN 690
  686 d = d-1: GOTO 630
  690 CALL coinc%: IF  NOT PEEK(cflag%) THEN 720
  700 spsize% = 449-spsize%: &    magnify or demagnify sprite
  710 POKE vreg%, spsize%: CALL wvr%
  715 d = 32: &    delay long enough to separate
  720 GOTO 630
```

Printer Ribbon Surgery

Contrary to what is implied on the packaging, an Adam ribbon cartridge is not reuseable. True it is multi-strike but what this really means is that more than one character is printed from the same spot on the ribbon before it is advanced. In fact, the SmartWriter printer strikes two characters per ribbon advance. Those of you who purchased your Adams primarily for its word processing capabilites have no doubt discovered that it doesn't take long to go through a printer ribbon. And to compound the problem further these ribbons are neither cheap nor easy to locate. You can of course mail order them through Coleco at $13.00 (excluding postage and handling) a shot. But this can be both a costly and lengthy process when you need to print out important documents urgently. After all, what good is having a word processor if you can't produce reports when you need them?

If you have tried to purchase a compatible ribbon in an office or computer supplies store, you may have noticed that although there are many types of ribbons (for typewriters and printers) none will fit the Adam printer as is. Well, if you give it further thought you will come to the same conclusion I did. Out of all those ribbons out there on the market, one has got to be similar at least in contents if not in shape. My assumption is correct and there are indeed many ribbons which are similar. Most are nearly identical except for the shape of the outer cartridge. With a little ingenuity and some physical labor, you will never need to rely on Coleco again as a supplier for printer ribbons.

Adam supposedly uses Diablo Hy-Type I multi-strike ribbons. If you can locate these great, otherwise do not discard your old ribbons when they are no longer good. Instead save them because I am going to describe how you can convert an easy-to-find ribbon (such as Diablo Hy-Type II or Qume I) into one that you can use on your Adam printer. The first thing you need to do is to remove the elastic belt from your old ribbon cartridge and by gently sliding a straight edge (such as a flat blade screwdriver) along its side open the ribbon housing. The top housing has six pins that slide firmly in six corresponding holes in the bottom housing. You must be careful not to force the two components open otherwise you risk breaking the pins. If you break all six pins you will not be able to reseal the two parts together. If you happen to break one or two pins you can still reuse the ribbon shell but it won't be as reliable. Locate each pin then using the straight edge as a lever, apply pressure near where the pin is seated in the hole. Do this at all six locations slightly lifting each pin out. Do not attempt to lift out one pin completely and then moving on to the next one. This way you will surely break one or more of the fragile pins.

Once you have opened the shell, carefully study the way the ribbon is routed before removing its contents. You might want to draw yourself a sketch the first time around. Remember that you will need the original left take-up spool, therefore remove and discard just the used ribbon it is holding. The next task is to

open the ribbon you purchased and transfer the complete right ribbon and hub assembly into the old cartridge. Carefully untape the beginning of the ribbon secured to the left spool, then tape it to take-up spool from your Adam ribbon. Finally, re-route the ribbon the way it was originally, reseal the shell, replace the outer black elastic belt and voila you are set to print in a matter of minutes. By making your own Adam ribbons, you can save yourself a lot of money because the ones you buy and put together yourself will cost considerably less.

Remember that you can always open the ribbon cartridge to fix any jammed or snapped ribbon. Splicing will also work nicely if you take your time and do a neat job. If not done properly however the left take-up spool will have difficulty winding the used ribbon. And here's another thing that you can try. Instead of replacing a used ribbon, rewind it. You'll find that the type is still pretty good, maybe not letter quality, but good enough for program listings and rough copies of text. To rewind, put a pen in the right hand (empty) spool and start spinning.

Review of FANTASY GAMER by Martin Consulting

In the last two issues, I reviewed Martin Consulting's 15 Basic Bonanza program. In that package I came across the text game MANSION, which proved to be enjoyable, as well as, frustrating. Upon solving Mansion, (after quite a few hours/ days) I thought quite highly of myself. Until Martin sent us FANTASY GAMER, and I am now back to step one.

Fantasy Gamer consists of two adventure games "with graphics" plus Adventure Creator which helps you design your own adventure game. Not bad on one TDK tape.

Basically, adventure games are like stories with you at the controls, that is, the story's outcome depends on your judgements and decisions. To play the game, you read the information about your position and status on the screen. As such, the game will ask you what you want to do and you must answer in TWO WORD commands, such as "go north" or "move rock", etc. Once the command is given, the game will take you into another situation and it is up to you to read carefully and pick up any clues and avoid red herrings.

It is definitely a good idea to keep some sort of record of your movements, so you don't backtrack. Set up a map in a grid fashion, that is, start your map with a rectangle (which represents the room you begin in) and add rectangles as you go along.

The two games in Fantasy Gamer are very similar but with a few differences. The first game, BOMB SQUAD is a "puzzle adventure" with only one solution. So, you must tackle this one logically to avoid death and red herrings. The second game, VISITOR, is a story with variable outcomes and more description and character development. So, the perspective of this game is to understand an opponent's personality rather than relying on logic and deduction.

Once you have gained some knowledge on text games, you can look

into the third program, ADVENTURE CREATOR, which teaches you how to create your own game. Martin certainly provides quite a bit of information to help you. They provide a general skeleton of the adventure program, explaining each section as you follow the program through. It is up to you to fill in the details of your own game, such as theme, time, characters, mapping your world, etc.. So let your imagination go.
Once the story is planned, you may start programming following the framework program provided. This program, as you go step by step will certainly increase your computer knowledge, as they have descriptions of parsers, sprites, and assembly language.
In conclusion, Fantasy Gamer will provide many hours of entertainment and frustration. GOOD LUCK.

REVIEW OF THE 64K MEMORY EXPANDER (BY HI-TEK MARKETING)

price $80.00

The 64K memory expander is not for everybody. In fact, even if you think that it is for you, it may not be. In order to decide this, let's look at what the memory expander can't do. First: it does not work in SmartBASIC at all. In word processing it increases the size of your workspace. However, because word processing files can only be as large as 6-7 pages (as recommended by Coleco Canada), this does not allow you to store larger sized files. If you want to print large documents that have been stored on separate files, the memory expander will allow you to dump more files into the work space than if you didn't have one. In Adamcalc, the memory expander acts as a printer buffer. This allows you to start working on another spreadsheet program as the printer prints out your first one. In CP/M, the memory expander is used as a RAM disk. Essentially it allows the user to store programs onto the chip as you would another drive. Of course you lose whatever you had when you shut the power off, but that's to be expected from volatile memory. The real advantage to a RAM disk is the speed that it gives to the storing and retrieving of information from it. It is instantaneous...or just about.

So why would you want to buy a memory expander? As you can see it does quite a lot of things. However, none of those things expand your memory. If you can live with this inconsistency, you can use the memory expander as described above. There may even be the possibility of getting some use out of it in Basic. If CP/M can use it as a RAM drive and Adamcalc use it as a printer buffer, there's no reason why Basic can't. Hopefully we'll see a development in this direction soon. By far the best application is that found with CP/M. You have to use it to appreciate it. And I'm sure that any Adam user who is getting tired of listening to his tapes whir will not take very long to appreciate this device.

Everything just said also applies to the original Coleco 64K memory expander. I have made a note, however, to review this particular one because of its price. Developed independently,

this version does everything that the Coleco one does. In fact, according to the manufacturer, it does them a little faster. Not that it matters too much at this speed anyways. But the price is less than one third that of Coleco who sell their expander for about $300.00. This is just too expensive for most users. This compatible version puts the memory expander within an acceptable price range for many more users, so I anticipate (and recommend) that you seriously think about purchasing it.
The memory expander is easy to install. It is a small circuit board that is placed in the third slot (right-most) in the memory console. The rough part faces left, the components right. The components are not protected or covered in a plastic shell in either version, so don't bend anything. The components look neater and flatter with this version so there is less to worry about in that regard. You have to be careful to guard the memory expander from any static electricity charge. Coleco provides an anti-static bag, Hi-Tek doesn't. Since you won't be removing it once installed, this oversight is not too large. It should be installed and, if necessary, removed with the power turned off.

PROGRAMS

```
   10 &          MIXNMATCH        04/02/85
   15 DIM da$(100), ff$(15), rt$(15), ac$(15)
   20 &    OPENING MENU
   24 &    COLOR SCREEN VIOLET
   25 POKE 17115, 13: TEXT
   30 NORMAL: HOME
   40 VTAB 3: HTAB 9: PRINT "MIX AND MATCH"
   50 VTAB 7: HTAB 3: PRINT "YOUR ADAM WILL BE SCRAMBLING           RANDOM WORDS
"
   60 PRINT
   70 HTAB 3: PRINT "IT IS YOUR JOB TO UNSCRAMBLE"
   80 HTAB 14: PRINT "THEM"
   90 VTAB 22: HTAB 6: INPUT "ENTER NUMBER (1 TO 999)        TO RANDOMIZE:"; x:
x = RND(-x)
  100 &          OFFER CHOICES
  104 &    COLOR SCREEN GREEN
  105 POKE 17115, 2: TEXT
  110 HOME: VTAB 3: HTAB 9: PRINT "MIX AND MATCH"
  120 VTAB 7: PRINT "YOUR CHOICES ARE:": VTAB 10
  130 PRINT "1)LEVEL ONE: SIMPLE!"
  140 PRINT " 2)LEVEL TWO: HARDER!"
  150 PRINT "  3)LEVEL THREE: EXPERT!"
  160 PRINT "   4)LEVEL FOUR: GENIUS!"
  170 PRINT "    5)LEVEL FIVE: WIZARD!"
  180 VTAB 18: HTAB 2: INPUT "WHAT IS YOUR CHOICE(1-5)"; ans
  185 &  CHECK TO SEE IF NUMBER IS VALID
  190 IF ans < 1 OR ans > 5 OR an <> INT(ans) THEN 180
  200 FOR i = 1 TO 100: READ da$(i): NEXT
  205 &       LEVEL ONE
  210 DATA  wish,fuel,kite,blue,were,knit,barn,poor,keen,peak,
  211 DATA  loom,gate,feed,seek,coat,rank,held,ride,desk,quit
  215 &       LEVEL TWO
  220 DATA  doubt,cruel,chief,thank,third,brief,altar,blank,trust,knife
  221 DATA   balks,paste,reply,excite,grant,grill,unite,enter,feast,quest
  225 &       LEVEL THREE
  230 DATA   border,ailing,pauper,stubby,yellow,encase,direct,abrupt,absolve
  231 DATA  headed,inquire,nudges,gimlet,sinful,slower,tunnel,victor,worker,voll
ey,reality
  235 &       LEVEL FOUR
  240 DATA   adjucate,euphenism,etiquette,scenario,vanquished,domesticate,circui
try
  241 DATA  headmaster,mimeograph,observance,tolerant,watershed,demolished,dunga
ree,favourable,chancellor
  242 DATA  toboggan,yielding,ostracized,pantomime
  245 &       LEVEL FIVE
  250 DATA  hereditory,controversial,patriarchy,vaporescence,dogmatism,circumspe
ct
  251 DATA  complimentary,adventure,obstructionist,potpourri,ferocious, accessor
ies,itinerary
  252 DATA  participate,holocaust,ideogram,reformation,introverted,loquascious,m
anipulate
  290 &       GET RANDOM WORD
  300 wo = INT(RND(1)*20)+(ans-1)*20+1
```

```
  305 &       SCRAMBLE IT
  310 wl = LEN(da$(wo)): FOR i = 1 TO wl: ff$(i) = MID$(da$(wo), i, 1): rt$(i) =
 ff$(i): NEXT i
  320 FOR i = 1 TO wl: ti = INT(RND(1)*wl)+1: rt$ = rt$(i): rt$(i) = rt$(ti): rt
$(ti) = rt$: ac$(i) = " ": NEXT i
  324 &    COLOR SCREEN BLUE
  325 POKE 17115, 7: TEXT
  330 HOME: VTAB 3: HTAB 9: PRINT "MIX AND MATCH"
  340 VTAB 7: HTAB 3: PRINT "ADAM HAS PICKED THE WORD"
  350 VTAB 16: HTAB 3: PRINT "YOU TRY TO UNSCRAMBLE IT."
  355 &       DRAW WORD BOX
  360 VTAB 10: HTAB 10: INVERSE: FOR i = 1 TO wl+4: PRINT " "; : NEXT
  370 VTAB 11: HTAB 10: PRINT " "; : HTAB 10+wl+3: PRINT " "; : VTAB 12: HTAB 10
: PRINT " "; : HTAB 10+wl+3
  380 PRINT " "; : VTAB 13: HTAB 10: PRINT " "; : HTAB 10+wl+3: PRINT " ";
  390 VTAB 14: HTAB 10: FOR i = 1 TO wl+4: PRINT " "; : NEXT i: NORMAL
  400 VTAB 12: HTAB 12: FOR i = 1 TO wl: PRINT rt$(i); : NEXT
  410 VTAB 18: HTAB 12: FOR i = 1 TO wl: PRINT CHR$(95); : NEXT i
  420 wp = 1
  430 &       INPUT GUESS
  440 GOTO 460
  450 &       THIS ALLOWS EDITING THE WORDS
  460 VTAB 18: HTAB 11+wp: GET ans$
  465 &       CHECK FOR LEFT ARROW
  470 IF ASC(ans$) = 163 THEN 560
  480 IF ASC(ans$) = 161 THEN 520
  485 REM     check for return,end of data mark
  490 IF ASC(ans$) = 13 THEN 620
  500 IF ans$ >= "A" AND ans$ <= "Z" GOTO 510
  505 IF ans$ >= "a" AND ans$ <= "z" GOTO 510
  507 ans$ = CHR$(95)
  510 ac$(wp) = ans$: PRINT ans$
  515 REM     move space  right
  520 wp = wp+1: IF wp > wl THEN  wp = wl
  530 GOTO 450
  540 IF ac$(wp) = "" THEN  PRINT CHR$(95); : GOTO 560
  550 PRINT ac$(wp);
  560 wp = wp-1: IF wp < 1 THEN  wp = 1
  570 GOTO 450
  580 IF ac$(wp) = "" THEN  PRINT CHR$(95); : GOTO 600
  590 PRINT acs$(wp);
  600 wp = wp+1: IF wp > wl THEN  wp = 1
  610 GOTO 450
  620 REM     this is the players guess
  630 FOR i = 1 TO wl: VTAB 12: HTAB 11+i: PRINT ff$(i); : IF ac$(i) = ff$(i) TH
EN  INVERSE: GOTO 650
  640 GOTO 660
  650 VTAB 18: HTAB 11+i: PRINT ac$(i); : NORMAL: wc = wc+1
  660 NEXT i: VTAB 20: HTAB 1: IF wc = wl THEN  PRINT "congratulations!": GOTO 6
80
  670 PRINT "SORRY YOUR GUESS WAS WRONG": PRINT "OUT OF "; wl; " LETTERS YOU HAD
 "; wc: PRINT " OF THEM CORRECT"
  680 INPUT "HIT RETURN TO CONTINUE "; ans$: RESTORE: wc = 0: GOTO 100



   10 LOMEM :29000
   20 DIM chord(5, 4), note(5, 5, 2)
   30 HOME: PRINT TAB(9); "tune generator"
   40 PRINT "please press <space bar> to"
   50 PRINT "begin. press <return> to stop"
   60 PRINT "THE program."
   70 IF PEEK(64885) <> 32 THEN 70
   80 FOR x = 28000 TO 28005: READ a: POKE x, a: NEXT x
   90 FOR x = 1 TO 5: READ chord(x, 1), chord(x, 2), chord(x, 3), chord(x, 4)
  100 FOR y = 1 TO 5: READ note(x, y, 1), note(x, y, 2)
  110 NEXT y: NEXT x
  120 FOR q = 144 TO 208 STEP 32: POKE 28006, q: CALL 28000: NEXT q
  130 t = INT(RND(1)*20)+20
  140 FOR x = 1 TO t
  150 r = INT(RND(1)*5)+1
  160 IF x = 1 THEN  r = 1
  170 IF x = t THEN  r = 4
  180 FOR z = 1 TO 4
  190 POKE 28006, chord(r, z): CALL 28000
  200 NEXT z
  210 FOR y = 1 TO 3: s = INT(RND(1)*5)+1
  220 IF PEEK(64885) = 13 THEN  FOR q = 159 TO 223 STEP 32: POKE 28006, q: CALL
28000: NEXT q: END
  230 FOR z = 1 TO 2
  240 POKE 28006, note(r, s, z): CALL 28000
  250 NEXT z
  260 FOR de = 1 TO 250: NEXT de: NEXT y: NEXT x
  270 FOR q = 0 TO 15 STEP .1: FOR n = 144 TO 208 STEP 32
  280 POKE 28006, n+q: CALL 28000
  290 NEXT n: NEXT q
  300 FOR q = 128 TO 192 STEP 32: POKE 28006, q: CALL 28000
  310 POKE 28006, 0: CALL 28000: NEXT q: GOTO 120
 1000 DATA        58,102,109,211,255,201
 2000 DATA        141,26,165,21,205,26,206,23,197,21,206,17
 2010 DATA        205,26,129,20,175,15,193,20,206,17,207,15
 2020 DATA        195,14,198,13,142,17,163,14,206,17,207,15
 2030 DATA        195,14,198,13,207,11,143,15,166,13,207,15
 2040 DATA        195,14,198,13,207,11,202,10,133,21,174,17
 2050 DATA        206,17,197,21,195,14,202,10,207,8
```

```
 0 LOMEM :29000
 1 toot = 144
 2 offtoot = 159
 3 GOSUB 40
 9 &
10 &              input sound parameters
11 &
12 HOME
13 FOR register = 0 TO 64 STEP 32
15 t = t+1: PRINT "tone:"; t
20 INPUT "pitch = "; pitch
21 GOSUB 100: NEXT register
30 INPUT "duration = "; duration
31 FOR register = 0 TO 64 STEP 32
32 GOSUB 160: NEXT register
34 GOSUB 170
35 FOR offtoot = 159 TO 223 STEP 32
36 GOSUB 180: NEXT offtoot
38 t = 0: RESTORE: GOTO 10
39 &
40 &              music assembler
41 &
50 FOR address = 28000 TO 28005
60 READ value
70 POKE address, value
80 NEXT address
90 DATA            58, 102, 109, 211, 255, 201
91 RETURN
99 &
100 &              main sound program
101 &
110 pitch = 3597000/(32*pitch)
120 second = pitch/16
130 first = 128+(pitch-(INT(second)*16))+register
140 POKE 28006, first: CALL 28000
150 POKE 28006, second: CALL 28000
151 RETURN
160 POKE 28006, toot+register: CALL 28000
168 RETURN
170 FOR delay = 1 TO duration*20: NEXT delay
178 RETURN
180 POKE 28006, offtoot: CALL 28000
181 RETURN
```

Computer Peripherals for the Adam

ADAMLINK DIRECT CONNECT MODEM (300 BAUD)	$159.99
ADAM COMPATIBLE 64K MEMORY EXPANDER	$80.00
EXTRA DIGITAL DATA DRIVE	$179.99
TRACTOR FEED	$130.00
ADDRESS BOOK FILER & AUTODIALER	$70.00

COMPUTER SUPPLIES FOR THE ADAM

DAISY WHEELS (GOTHIC 15, SCRIPT 12, LIGHT ITALIC 12, LETTER GOTHIC 12, MANIFOLD 10)	$10.00 each
PRINTER RIBBON	$10.00
ADAM COMPATIBLE DATA PACKS ... $6.00 each, 5 for $28.00 or 10 for $53.00.	
ADAM MONITOR CABLE	$14.99
NO NAME DISKETTES ..... 10 for	$17.00
FUJI DISKETTES ..... 10 for	$26.00

COLECO SOFTWARE FOR THE ADAM

CP/M 2.2 & ASSEMBLER (DP,DISK)	$95.00
ADAMCALC (DP) SPREADSHEET PROGRAM	$61.00
SMARTLOGO (DP) EDUCATIONAL PROGRAM	$98.00
SMARTFILER (DP,DISK) ALL PURPOSE FILER ...	$45.00
EXPERTYPE (DP,DISK) ELECTRONIC TYPING AID	$55.00
SMART LETTERS AND FORMS (DP,DISK)	$45.99
ELECTRONIC FLASHCARD MAKER (DP) EDUCAT'AL	$45.00
FLASH FACTS-TRIVIA (DP) EDUCATIONAL	$25.00
FLASH FACTS-VOCABULATOR (DP) EDUCATIONAL	$25.00
ELECTRONIC WORDBOOK (DP) EDUCATIONAL	$48.00
RECIPE FILER (DP,DISK)	$45.00
DONKEY KONG (DP)	$45.00
DONKEY KONG JR (DP)	$45.00
ZAXXON (DP)	$50.00
DRAGON'S LAIR (DP,DISK)	$50.00
FAMILY FEUD (AVAILABLE SOON)	$45.00
JEOPORDY (AVAILABLE SOON)	$45.00

***** PERSONAL ACCOUNTANT *****
3 FINANCIAL PROGRAMS IN ONE.
SIMULTANEOUSLY POSTS ENTRIES,COMPILES FINANCIAL REPORTS,LARGE DATA BASE,BREAKS DOWN EXPENSES,AMORTIZATION/PAYMENT SCHEDULES .. $55.00

Send orders to: HI-TEK MARKETING
244 PENBROOKE CLOSE S.E.
CALGARY, ALBERTA, T2A 3P1
(403)-272-3927