SYNTAX 2.2

MAY JUNE 86

T A B L E O F C O N T E N T S

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

POSTAGE PAID
Mtl. PQ
Permit Pending

E-Z COPY

E-Z COPY is a self-loading program that allows you to make copies of your original tapes or disks. The program was written in Basic but the Basic interpreter has also been modified. For this reason, we do not recommend that you store anything on the E-Z COPY tape or disk. Just use E-Z COPY to duplicate your programs.

HOW TO USE E-Z COPY

Insert the E-Z COPY media in your first drive and pull the computer reset button towards you. The screen will turn black and then red with the FCAUG logo held on the screen for a short time. The second screen then asks you how many blocks you want to copy. Suggestions for tape and disk length copies appear on screen at this stage. When the cursor is flashing beside "start block" type 0 (zero) and hit return. The cursor then flashes beside "last block". Type in the number of the last block to be copied. If it's tape to tape, type in "255" and hit return. If you are copying to a disk (or from, for that matter) type in "159" and hit return. Make sure you don't ask for more blocks that exist on your source or destination. You'll be telling the program to do something that it simply can't do. The next screen asks you from where you want to copy (source) and to where you want your copy made (destination). Type in the number that corresponds to the drive(s) that you'll be using for the source and then for the destination. For single drive copies this number will be the same.

Once the copy parameters are defined as above, check to see that your media is in the appropriate drive(s). You then press any key to start the duplication process. If you are using one drive to back up your media the program will stop every 20 blocks read and every 20 written. This allows you to change between source and destination media.

After you have made a copy you have the option of quitting the program or making another copy. If you press "y"es the screen will freeze and in a few seconds the program will return you to the first input screen.

SOME TIPS ON THE USE OF E-Z COPY

As we have mentioned in the past, there are other copy programs available. Our program is different in that it allows you to choose which blocks you want to copy whereas most programs make you copy the entire tape or disk even though 3/4 of it may be blank. Because of this feature, you can save a lot of time by copying just the blocks that you need. For example, the entire SmartBasic can be copied by choosing "0" as your start block and "29" as your last block.

The following is a short list of programs indicating their specific last block to copy. The first number is in decimal, the second is in hexadecimal. (You may need to use the hex number in

some copying programs. E-Z COPY uses the decimal number.) The start block for each program is always zero (0). When E-Z COPY asks you for a start block, enter zero (0). When asked for the lask block, consult the following table and then enter the decimal number next to the program that you wish to copy. By using this list you won't have to copy all the blocks from your source, but just the ones you need. This should optimise your copying time.

Program	Last Block (dec, hex)
SmartLOGO:	127 (7F)
Coleco Home Library:	127 (7F)
Tax Planner:	63 (3F)
AdamLink 2:	63 (3F)
SmartFiler:	67 (43)
SmartBASIC:	29 (1D)
SmartBASIC II:	50 (32)
AdamCalc:	127 (7F)
CP/M 2.2:	159 (9F)
Flashcard Maker:	57 (39)
Expertype	119 (77)

There are some things that you should watch out for when making a tape to disk copy. Here are 3 examples. Some programs (Smart Letters and Forms, for example) are longer than 160 blocks. Since a disk only has 160 blocks, you can not make a straight tape to disk copy of these types of programs. SmartBasic has to be modified after it is copied to disk. See SYNTAX 1.3 for how to do this. And lastly, most game packs cannot be copied to a blank tape. They can sometimes be copied to a disk but extensive modifications are always necessary. They can be copied over an old game pack, however. These limitations with game packs are due to a difference in the way that game packs and all other tapes are formatted. Thus E-Z COPY will not work with game packs.

WHY E-Z COPY?

The answer is quite simple. In a word, self-sufficiency. Computer tapes and disks wear out. Accidents happen and tapes get erased When this occurs, you either pay through the nose for another copy from Coleco, or as may happen shortly programs may not be available at all. Without a doubt replacements can be gotten from FCAUG or AEC but why bother, when with a little foresight and prevention you can provide yourself with all the backups you want with E-Z COPY.

E-Z COPY is available from FCAUG for just $15.00. Blank tapes are available for $10.00 each. Blank formatted disks are available for $2.00 eack. Write us at our regular address.

ADAMCalc: Monthly Interest Calculation and Analysis

Contributed by Michael Darracott

Adam owners who also have acquired the ADAMCalc program and have tried to develop an amortization schedule for their own house mortgage have no doubt found that their calculations differ from the schedule used by their mortgagor or that a balance quoted by the mortgagor on request also differs from their calculations. Take heart - there is nothing wrong with your computer, ADAMCalc or you! The problem lies in the difference between the American and the Canadian methods of calculating interest. The American method is the basis of the material found in your ADAMCalc manual on pages 118-123. Looking at page 119, you will find that the periodic interest referred to is simply the annual rate divided by 12 (in the case of monthly payments). This is probably the method you learned in high school. The sample schedule shown on page 122 indicates that the first monthly payment is made up of interest of $40.00 (calculated as 1/12 X .12 X $4,000.00). This is deducted from the monthly payment of $188.29, leaving $148.29 to apply to principal and reducing the loan amount to $3,851.71.

In Canada, we are affected by an old federal statute - The Interest Act. That Act was passed in the days when Canada was primarily an agricultural economy. Since it was customary for farmers and even city-dwellers to make their mortgage payments annually or at best semi-annually and to pay a principal amount on top of the interest charged, it was a relatively simple matter to calculate the interest. With the advent of so-called blended mortgage payments (i.e. payments which contain a mix of interest payment and principal reduction), it was required by the Interest Act, probably because of the difficulty the average person would have in calculating interest and the possibility of fraudulent practices creeping into the system, that the interest rate must be expressed in terms of the effective annual or semi-annual interest rate. Hence, the amortization schedules used by many mortgage companies and published by such companies as Boston Publishing and other Canadian firms use a monthly factor which equates to the stated semi-annual interest rate. Mortgages usuallly state interest on a semi-annual instead of annual rate because semi-annual gives the mortgagor (the lender) a slightly higher yield. The formula to arrive at the equivalent monthly factor to calculate your mortgage schedule is as follows:

$$\sqrt[6]{\left(1+\frac{\text{annual interest rate}}{2}\right)} - 1$$

EXAMPLE - A Canadian mortgage in the amount of $10,000 calls for interest at the rate of 8%, calculated and compounded semi-annually, not in advance, with monthly payments of $76.33. The interest for each month is .0065581970 (the sixth root of one plus 1/2 the annual rate less one). Using the above formula:

$$\sqrt[6]{(1+\frac{.08}{2})} - 1$$

Refer to the column headed "Canadian Effective Monthly Rate" next to 8% on the accompanying comparison table and you will find the monthly equivalent to be .0065581970. To prove its accuracy, take that figure plus one to the power of 6 (i.e. multiply the number plus one by itself 6 times and then deduct one). You can do this using a simple pocket calculator or the one built in the ADAMCalc program. If you use the former, enter 1.0065581970 press the multiplication button and then the equal sign five times (always one less than the power) and you will come up with 1.04 or very close to it (mine says 1.0399991). Take away the "1" and you are left with .04 or one half the annual rate of 8%.

It might be of interest to know that some sophisticated lenders, with powerful computers, use a daily compounding of interest. The daily interest rate is the one hundred and eighty second and one half root of one plus one half the annual rate. Each time a payment is applied the interest is calculated from the time the last payment was made by taking the daily factor to the power of the number of days to today, (with the balance of the fixed payment being applied to the principal balance). This way the lender (mortgagor) is getting interest on each dollar for each day the dollar is outstanding and the borrower (mortgagee) is getting the benefit of paying less interest if he pays early - a very fair system.

Proof: $\sqrt[182\ 1/2]{1+\frac{\text{annual rate}}{2}} - 1$

is the daily factor. The interest for a half year would be:

$$\left(\sqrt[182\ 1/2]{1+\frac{\text{annual rate}}{2}} - 1\right)^{182\ 1/2}$$

or cancelling out the root and power signs, we are left with

$$1+\frac{\text{annual rate}}{2} - 1$$

But the fact remains that the majority of lenders, private and corporate, still use the amortization schedule even though it has shortcomings. For example, it assumes that the year is split into twelve equal periods each of the same length which of course fails to recognize that some months are longer than others. The differences are small however and tend to even themselves out.

In summary, referring once again to the schedule on page 122 of the ADAMCalc manual, you can show the annual interest rate (using "Free Format" to accommodate the long string following the decimal) as the right hand column on the comparison table and

then divide by 12 to come up with the monthly interest factor to use in Canada. You could also use the monthly factor from the second column from the right. Omit the division by 12 and instead multiply the previous principal balance by it, deduct the interest amount from the monthly payment and reduce the principal balance by that difference. This would however require making unnecessary changes to the program.

One interesting thing to note is that the effective annual rate in Canada is slightly less than the rate under the U.S. method. This is due to the fact that the American method has a more frequent period of compounding (monthly as opposed to semi-annually) than the Canadian method. I hope that this discussion will be of some help to you in figuring out your mortgage. While every effort has been made to ensure the accuracy of the figures and calculations presented in this article, no responsibility is assumed for any matters arising out of the use of the material.

MONTHLY INTEREST FACTORS

Interest for one month at nominal annual rates shown, based upon interest compounded semi-annually.

8 % – .006 558 1970	10⅜% – .008 464 6722
8⅛% – .006 658 9889	10½% – .008 564 5152
8¼% – .006 759 7303	10⅝% – .008 664 3089
8⅜% – .006 860 4214	10¾% – .008 764 0532
8½% – .006 961 0622	10⅞% – .008 863 7482
8⅝% – .007 061 6527	11 % – .008 963 3940
8¾% – .007 162 1929	11⅛% – .009 062 9906
8⅞% – .007 262 6831	11¼% – .009 162 5381
9 % – .007 363 1231	11⅜% – .009 262 0365
9⅛% – .007 463 5130	11½% – .009 361 4858
9¼% – .007 563 8530	11⅝% – .009 460 8863
9⅜% – .007 664 1431	11¾% – .009 560 2378
9½% – .007 764 3832	11⅞% – .009 659 5404
9⅝% – .007 864 5735	12 % – .009 758 7942
9¾% – .007 964 7141	12⅛% – .009 857 9993
9⅞% – .008 064 8049	12¼% – .009 957 1557
10 % – .008 164 8461	12⅜% – .010 056 2634
10⅛% – .008 264 8377	12½% – .010 155 3225
10¼% – .008 364 7797	12⅝% – .010 254 3331

Interest for one month on any amount may be obtained by multiplying that amount by this factor.

This chart also shows the effective monthly interest rates used in Canada. Source - Financial Publishing.

INTEREST RATE EXPRESSED AS %	DECIMAL	U.S. ANNUAL RATE	U.S. EFFECTIVE MONTHLY RATE(3)	CANADIAN ANNUAL RATE(1)	CANADIAN EFFECTIVE MONTHLY RATE(2)	CANADIAN X 12
8	.08	8	.00666666	8	.0065581970	.078698364
8 1/8	.08125	8 1/8	.00677080	8 1/8	.0066589889	.079907867
8 1/4	.0825	8 1/4	.00687500	8 1/4	.0067597803	.081116764
8 3/8	.08375	8 3/8	.00697916	8 3/8	.0068604214	.082325057
8 1/2	.085	8 1/2	.00708333	8 1/2	.0069610622	.083544746
8 5/8	.08625	8 5/8	.00718750	8 5/8	.0070616527	.084739832
8 3/4	.0875	8 3/4	.00729160	8 3/4	.0071621929	.085946315
8 7/8	.08875	8 7/8	.00739583	8 7/8	.0072626831	.087152197
9	.09	9	.00750000	9	.0073631231	.088357477
9 1/8	.09125	9 1/8	.00760410	9 1/8	.0074635130	.089562156
9 1/4	.0925	9 1/4	.00770830	9 1/4	.0075638530	.090766236
9 3/8	.09375	9 3/8	.00781250	9 3/8	.0076641431	.091969717
9 1/2	.095	9 1/2	.00791666	9 1/2	.0077643832	.093172598
9 5/8	.09625	9 5/8	.00802080	9 5/8	.0078645735	.094374882
9 3/4	.0975	9 3/4	.00812500	9 3/4	.0079647141	.095576569
9 7/8	.09875	9 7/8	.00822910	9 7/8	.0080648049	.096777659
10	.10	10	.00833333	10	.0081648461	.097978153
10 1/8	.10125	10 1/8	.00843750	10 1/8	.0082648377	.099178052
10 1/4	.1025	10 1/4	.00854166	10 1/4	.0083647797	.10037736
10 3/8	.10375	10 3/8	.00864580	10 3/8	.0084646722	.10157607
10 1/2	.105	10 1/2	.00875000	10 1/2	.0085645152	.10277418
10 5/8	.10625	10 5/8	.00885410	10 5/8	.0086643089	.10397171
10 3/4	.1075	10 3/4	.00895830	10 3/4	.0087640532	.10516864
10 7/8	.10875	10 7/8	.00906250	10 7/8	.0088637482	.10636498
11	.11	11	.00916666	11	.0089633940	.10756073
11 1/8	.11125	11 1/8	.00927080	11 1/8	.0090629906	.10875589
11 1/4	.1125	11 1/4	.00937500	11 1/4	.0091625381	.10995046
11 3/8	.11375	11 3/8	.00947910	11 3/8	.0092620365	.11114444
11 1/2	.115	11 1/2	.00958333	11 1/2	.0093614858	.11233783
11 5/8	.11625	11 5/8	.00968750	11 5/8	.0094608863	.11353064
11 3/4	.1175	11 3/4	.00979166	11 3/4	.0095602378	.11472285
11 7/8	.11875	11 7/8	.00989166	11 7/8	.0096595404	.11591448
12	.12	12	.01000000	12	.0097587942	.11710553
12 1/8	.12125	12 1/8	.01014100	12 1/8	.0098579993	.11829599
12 1/4	.1225	12 1/4	.01020833	12 1/4	.0099571557	.11948587
12 3/8	.12375	12 3/8	.01031250	12 3/8	.0100562634	.12067516
12 1/2	.125	12 1/2	.01041666	12 1/2	.0101553225	.12186387
12 5/8	.12625	12 5/8	.01052080	12 5/8	.0102543331	.123052
12 3/4	.1275	12 3/4	.01062500	12 3/4	.0103532952	.12423954
12 7/8	.12875	12 7/8	.01072910	12 7/8	.0104522088	.12542651
13	.13	13	.01083333	13	.0105510740	.12661289
13 1/8	.13125	13 1/8	.01093750	13 1/8	.0106498909	.12779869
13 1/4	.1325	13 1/4	.01104166	13 1/4	.0107486596	.12898392
13 3/8	.13375	13 3/8	.01114580	13 3/8	.0108473799	.13016856
13 1/2	.135	13 1/2	.01125000	13 1/2	.0109460522	.13135263
13 5/8	.13625	13 5/8	.01135410	13 5/8	.0110446762	.13253611
13 3/4	.1375	13 3/4	.01145833	13 3/4	.0111432522	.13371903
13 7/8	13875	13 7/8	.01156250	13 7/8	.0112417802	.13491336
14	.14	14	.01166666	14	.0113402602	.13608312

NOTES: (1) Rate as shown in Canadian mortgage expressed as % calculated and compounded semi-annually, not in advance.
(2) Monthly rate which equates to (1) above
(3) U.S. annual rate divided by 12

BASIC PARTS - A look at Default Drives and the Disk Manager

The first thing most users accustomed to working with a single data drive discover when first connecting their disk drive or a second data drive, is that they cannot access it. What they fail to realize is that under SmartBASIC, the drive the Adam is ready to accept input or output from is the first data drive. This is referred to as the default drive. Since the default is always data drive #1, when we are working with two or more drives, we need to tell Adam when we want to access a different one. To do this we use the following labels: D1, D2, D5, D6. These are for the no. 1 and 2 data drives, and the no. 1 and 2 disk drives respectively. For example, to LOAD a file called "flag" stored on a disk inserted in the #1 disk drive, you must enter D5 after the filename (LOAD flag D5). This tells Adam to go read the disk and LOAD a file called "flag". If this file was stored on a data pack which you had inserted in the second data drive, you would simply substitute D2 for D5. Once you have indicated a specific drive, it becomes "logged-in" until you change it. In the previous example, disk drive #1 would now act as the default drive until you indicate that you want to use another.

When you connect and use a disk drive, you must remember to always switch it on before you power on your computer otherwise Adam will not recognize it. If you have a power outlet bar with its own switch, you can turn on both computer and disk drive(s) simultaneously. The labels for accessing the disk drives change depending on which software you are using. Refer to the reference card that came with the disk drive for the correct label. With some software (e.g. SmartWRITER, SmartFILER) access is achieved by use of the smart keys. Also remember that some software like the internal word processor can only access one disk drive.

The SmartBASIC program on your tape sets the default drive to D1 every time you boot it. It then activates data drive #1 to search for and RUN a file named HELLO. If no such file exists, it just waits for you to enter the first command. If you load your Basic from disk and there is no tape in data drive #1, it will spin momemtarily and make a whirring noise. This is because when you make a straight tape-to-disk copy of SmartBASIC, the program still behaves the same way. The fact that it has been transferred to a disk does not change anything. After your first command you must enter the drive number (D5) because otherwise it always assumes data drive #1. With this article we include a program submitted by a fellow user which offers the simplest and quickest solution to modifying SmartBASIC to run from disk. After you have made a copy of SmartBASIC V1.0 on disk using E-Z COPY or some other program, run this short program, insert your disk, and press the STORE/GET key. Now every time you load your SmartBASIC disk the default drive will be set as disk drive #1. If you have a HELLO file on the same disk, it will also work. SmartBASIC V2.0 does not have this problem. It will set the default drive as the drive from which it was booted. This is just one of the many improvements it offers over the original version.

The Disk Manager program included with the disk drive has several important functions. First and foremost, it allows you to format standard disks for use with the Adam disk drive. In the FORMAT mode, you can format the disk as either a System Disk or a Data Disk. There is a big difference between the two. The former always puts the Disk Manager program on the disk being formatted. This naturally reduces the storing capacity of the disk. Use the System Disk option only when you want to include the Disk Manager program on your disk or to make an extra copy of it. To obtain maximum storing capacity from each disk, format them as Data Disks.

The Disk Manager program allows you to perform four other useful file maintenance functions: RENAME, RUN, COPY, and DELETE. There is nothing different about these except for the RUN option. This command allows you to run your personal programs plus SmartBASIC, SmartFILER, and others using the Disk Manager. The advantage of this method over the usual method of loading most likely involves the EOS. The EOS file appears on the same disk. EOS is short for Elementary Operating System which seems to be a revised version of Adam's operating system. Every time we boot the Disk Manager, it also loads automatically and it appears that we are actually upgrading the internal system. By running SmartBASIC using the Disk Manager, we are therefore loading it under the revised system. This might not seem like much, but it might help to avoid some of the operating quirks often encountered with some Adams.

```
  5 &  FIXHELLO
 10 &  Submitted by Mike Goheen
 15 &
 20 &  A program to fix the HELLO function
 30 &  after you copy SmartBASIC to disk
 40 &  Use ONLY on SmartBASIC v79!
 50 &  To check if your version of SmartBASIC is 79
 60 &  Type PRINT PEEK(260) and hit the <RETURN> KEY
 70 &  Run the program, insert your disk, press <STORE/GET>.
 80 &  Each time you boot your SmartBASIC Disk, the HELLO program
 90 &  will run automatically and the default drive set to D5
100 &  Source: Family Computing Forum on Compuserve
110 &  Contributed to Library by Uncle Ernie [72167,3241]
120 LOMEM :31100
130 HOME
140 PRINT "SET DISK #1 as DEFAULT DRIVE"
150 PRINT "FOR SmartBASIC BOOT-UP.....": PRINT: PRINT: PRINT
160 PRINT "INSERT SmartBASIC DISK,"
170 PRINT "PRESS <STORE/GET> KEY": GET a$
180 IF a$ <> CHR$(147) THEN INVERSE
190 PRINT CHR$(7); "  COMMAND CANCELLED  "; cal: NORMAL: END
200 FOR i = 28000 TO 28014: READ x: POKE i, x: NEXT
210 CALL 28000
220 POKE 30001, 4: POKE 28012, 246
230 CALL 28000
240 INVERSE: PRINT CHR$(7); "      COMPLETED      ": NORMAL: END
250 DATA 62,4,33,48,117,17,18,0,1,0,0,205,243,252,201
```

Machine Language Primer

Welcome to the first in a series of articles that will introduce you to machine language programming and the architecture of the Adam. This will allow us to make use of a whole bunch of system routines which simplify our programming quite a bit.

I'll start by explaining the difference between assembly language and machine language. The Adam's brain is the Z80 central processing unit (CPU). It's job is to execute a series of 8 bit numbers. What these numbers are and more inportantly, what they do when executed is determined by the CPU manufacturer. In our case the Zilog Corporation is responsible for designing the Z80 chip and the instruction set that goes along with it. (Yes, I know that the Z80 is based on Intel's 8080 chip, but that's anothe story!). These numbers represent simple operations within the CPU. Since a sequence of numbers is meaningless to most humans, the numbers are most often represented by 3 or 4 lettered names (called mnemonics since it tells what the machine code number does). AS soon as we program using the mnemonics instead of the numbers then we are programming in asssembly language. But for the Z80 to understand, our assembly language programs have to converted into machine code. A program that does the conversion is called an assembler. Note that an assembler does a simple one to one conversion of mnemonics to machine code numbers. If we do the translation by hand then we are programming in machine language.

An assembler mnemonic or instruction is limited in what it can do. By combining a bunch of these simple instructions together we can do very powerful things. In fact machine code is the basis for every computer programming language. For example, one BASIC statement eventually gets translated into a wole bunch of machine code numbers. Our view of programming is quite different in assembly language because of its "lowlevelness" than most computer languages. One might say it is a bit more tedious to program in it (and alot more would probably say it was much more tedious!). If you can approach machine language programming like you would a puzzle you may find it quite enjoyable. The reason programmers descend to machine language is usually due to necessity. It is always the fastest way for the computer to do something and many times it is the only way. ie. think of how we get sound, sprites, etc. out of SmartBASIC.

The Adam's Z80 CPU is a relatively old microprocessor being introduced in 1976 (remember that when talking about computers, time is small while everything else is huge!). It is a suprisingly fast and powerful CPU for its size and age, though. Like all CPU's this one has a set of registers. Registers can be thought of as special memory locations. They are special because you access them by name and they are much faster than a regular memory location since they are inside the CPU. Probably the most important reason that they are special is because the CPU uses them (explicitely or implicitely) when executing an instruction.

The Z80 has a group of so called general purpose registers. By this it is meant that you can use them for almost anything.

Probably the most useful and general register is the A register. This is also called the accumulator since many instructions use this register in doing calculations. The other 8 bit general purpose registers are the B, C, D, E, H, and L registers. What makes these special is that they can be paired off to create single 16 bit registers. These 16 bit registers are referred to as BC, DE, and HL. Some instructions implicitely use these 16 bit registers. The DE and HL registers are often used to point to a memory location (as in tranferring bytes from one location to another) while the BC register is used as a data counter. The last register in this set is the F or flag register. Each bit in this register will change according to a particular condition. Some possible conditions may be a byte overflowing as a result of an arithmetic operation on it or the most significant bit in a byte becoming a one. When an instruction is executing it may implicitely set (make a one) or reset (clear it to zero) a paricular bit in this register. Some instructions may check a bit in this register to branch on it if the condition it's looking for occurred. The bits are called flags and are the following:

C (carry) flag - this is set when an operation is done which results in a value which is too large to fit in a byte.

Z (zero) flag - this is set when the previous result was zero.

S (sign) flag - set when the most significant (leftmost) bit of the result is one.

P/V (Parity or Overflow) flag - set if the result contains an even number of ones or if it overflows.

H (half carry) flag - same as the C flag except looks at the 4th bit rather than bit 7 (most significant bit).

N (add-subtract) flag - set if the operation is a subtraction.

The Z80 designers provided us with two identical sets of general purpose registers. You may only access one set at a time by switching between the two. This is known as context switching and was a neat concept for microprocessors at the time but is not of great importance to you or me.

There are also a bunch of special purpose registers, that is the CPU makes use of most of these. The 8 bit R (memory refresh) register is used to speed up the fetching of the next instruction in memory and shouldn't be changed by us. The I (interrupt) register is used to indirectly access any memory location when handling an interrupt. Interrupts are a more advanced subject so I will pass on explaining them for now. There are two 16 bit index registers, IX and IY, which can be used to implement arrays in machine language. Another 16 bit register is the stack pointer (SP) register which is used to implement a stack. The stack is used alot by the CPU to store return addresses but can also be used by us to store our own data. It is good for passing data between subroutines. The last 16 bit register is the program counter (PC) register. It points to where the CPU will fetch the next instruction and should not be modified by us.

This article has introduced us to the idea of machine code and to the innards of the Z80. In the next issue I hope to have a mnemonic to machine code table for the entire Z80 instruction set.

This will become valuable when we start writing our machine code routines. See you then.

CP/M CORNER

Sooner or later you are going to get tired of using CP/M with its standard blue screen, welcome message, and all the rest. When that day comes you'll be flipping to the section in your CP/M manual that discusses the command "CONFIG". There you'll find that this command allows you to redesign the look of your CP/M screen any way that you want it to look. You can change the screen color, charactor color, cursor color, cursor shape, and the state of the Smart keys for when CP/M is initially booted.

The CONFIG main menu provides you with 6 options. The first exits the program. The next 2 read the editable tables of options from either memory (the present disk) or another drive. The next 2 options are the opposite of the above, they are the write options. And the last option allows you to modify the tables that you read in. Obviously this is the one we're interested in. Without it we could only read in a set of screen characteristics and write them out again to another disk. But we want to make our own modifications, so we use the editting option after reading in the tables.

There are 9 options on the CONFIG edit menu, we will concern ourselves with half of them (the first option is the return to main menu option). These are options 4,5,6,and 9 and are the features that I referred to in the first paragraph. I could look at what these options can do for you but that wouldn't be too interesting. Besides the changes that you are able to make are discussed at length in the CP/M manual. Therefore, in order to make this article more interesting, I will pay particular attention to where and how these changes are made on the actual CP/M disk.

Most of the tables for CONFIG are located in the second block of the CP/M disk, ie the first block of the system tracks. As we discussed last issue, this part of CP/M is normally inaccessible to the user. By making the changes to SYSGEN as described in SYNTAX 2.1, and then storing SYSGEN and the system together as CPM1.COM (or whatever name you wish to call it, you can use DDT to make the changes you wish directly to the system. In this particular case we will make specific changes to block 1 thereby mimicing the effect of CONFIG. In this way, we won't be doing anything that CONFIG can't do (although, later we will learn how to change the CP/M Welcome message), but we will be learning how these changes are made. I trust that this information is enough to justify this article.

First we start with a correction. The number under NEXT after you write "ddt sysgen.com" should be 0700 and not 0800. The number under NEXT is used in determining what number you use when you "SAVE" a memory image into a file. The CP/M manual tells you how to determine how many pages to save by telling you to look at the

2 leftmost digits under NEXT, converting them to decimal and then using them in the command. This works but if the 2 rightmost numbers are 00 then you end up saving an extra page each time you play with this file. And of course, this is the case with sysgen and cpm1. (If all these units of storage - blocks, pages, etc is confusing, don't worry, at the end of this article I'll give a quick breakdown of what is what.)

After you made your changes to sysgen you saved it using 8 pages , ie "save 8 sysgen.com". But according to the above discussion, you would only have to save 6 pages. Not to worry, if you used 8 instead of 6, it doesn't really matter in this and most other cases. Whether 2 extra pages of memory are saved with sysgen or not is irrelevent. Sysgen stops executing after 6 pages and places the system starting at the 9th page. So the 2 pages (7 and 8) in between, whether put there with the sysgen or the left over section of a previous program will not concern us nor will it affect the program in any way. I mention this because when going through CPM1.COM with ddt you'll recognise the sysgen and system parts but not pages 7 and 8. As long as all of sysgen and the system is there you have nothing to worry about; in regard to the aforementioned blocks, ignore them.

With that out of the way, write "ddt cpm1.com". NEXT will have a 3D00 under it, this converts to 61 00 decimal, your PC is at 1 00, so you save (when we're finished editing) the file with a 60, as per SYNTAX 2.1. Check to see that SYSGEN is from 100 to 6FF. The contents of 700 to 8FF are irrelevent. The system starts at 900. From 900 to CFF is block 0, the boot block. From D00 to 11FF, we find the block we want, block 1. This is the block that contains most of the tables that CONFIG normally accesses.

Look at address 0E5F. If, upon booting CP/M, your Smart keys are on, you will find a 1A here. Changing this number to a 19 will tell the computer that you do not want the Smart keys on when you boot up CP/M. You should know how to performeditting changes with ddt by now but I'll show you once more. Type the address you want to change. Type "se5f". Adam will print 1A. Beside it you should type 19. The next address and its memory contents will then be displayed on the next line. This will be 0E60, contents 1B. You want to exit, not edit so hit the period key, followed by <return>. If that's all the changes you want to make hit ^C or type g0 to exit ddt. After saving the editted CPM1, under a name like CPM2, execute the program and write the altered system back to your system tracks. Booting CP/M will display your change.

We make other changes similiarly. The byte for cursor color is at 0F91. Most probably it is a 05 (blue). Change this number to any number from 00 to 0F. (Follow the table found on the left side of p.14 SYNTAX 1.1 for all color changes, bot remember to change "black" to "transparent" for color number 0 or you can look at the color menu on p. C47 of the CP/M manual.) The shape of the cursor can also be changed. The bytes that store the cursor shape are from 0F93 to 0F9A. Each byte represents a cross section of the cursor from start to bottom. This gives you an 8 X 8 grid to work on when you design your curs r shape. Follow the "bit map" on p

C47 to get an idea of how to design your cursor shape. When you understand that look at how these 8 bytes are stored on the disk. You will notice that the 8 entries in the bit map correspond directly with the 8 bytes here. The 8 bytes are simply the hexadecimal representation of the binary bit map entries.

Right after the cursor shape area are 2 bytes (0F9B and 0F9C). The first defines the color for the characters that are printed on the screen. The second provides the inverse color of any charaters. The last change that I'll look at is for the screen or background color. This byte is stored in block 2 of the CP/M disk so it is a little further on in our CPM1 file. Block 2 goes from 1200 to 16FF. The screen color byte is at 1227. The contents of these areas will be from 00 to 0F. Follow the aforementioned color charts to get the color that you want.

Adam Graphics: Shape editor

This article will explain a shape table editor program as well as expose a discrepancy of plotting vector codes.

As promised in the last issue, I have written a shape table editor. This program will allow you to draw your shapes on the screen one at a time. When you have finished drawing your last shape, the assembled shape table will be printed out for you. This takes all the hard work out of using shape tables in your programs. The program makes use of the low resolution graphics mode to draw your shapes. As you recall this gives you a 40 by 40 grid of blocks and a 4 line text window at the bottom of the screen. When drawing your shape you may either turn the dots on or off. Dots that will be off are represented as purple or black blocks while those that will be on are white. The background is black, allowing you to see your plotting motion. The program provides instructions at the bottom of the screen but I will go over the editing process in more detail here.

You will be presented with a blank drawing area which measures 40 by 40 blocks. This means your shapes can be no higher or longer than 40 dots. You start by moving the flashing cursor to a part of the screen that will allow you enough room to plot your shape. Pressing the zero key on the keypad will anchor the cursor at its starting position. You may then begin drawing your shape on the screen. To change the cursor color and hence the plotting, you press one of the fire buttons. If you make a mistake you can press the * key on the keypad to move the cursor back to the previous position. Doing this several times will bring you back to the cursor starting position. When you are satisfied with your shape you can move on by pressing the # key on the keypad. This will give you the option of creating a new shape for your shape table or ending the table. If you choose to end the table you will have the completed shape table printed out (either on screen or on the printer). If you want to continue on another shape then you will be given a new blank editing area. An important thing to note when drawing is that the flashing cursor is where the next point will be plotted and shouldn't be viewed as part of your shape. That is,

note what was originally under the cursor position ; if it was white then the point will be turned on, if it was black or purple then it will be turned off. Also the editor will allow you to only move the purple cursor up one dot at a time. This is due to the constraints placed on moving up without plotting (see previous article).

The program spends most of its time trying to place the 3 bit plotting vector code in the appropriate spot in the appropriate byte. A major problem I discovered has to do with the move up without plotting vector code (bits 000). A byte of the following form:

00 000 XXX

where the series of X's represents at least one 1 bit will not work as you expect. Refering to the plotting vector codes given in the last article we see that the second plotting code says to move up without plotting. In actual fact the last two zero codes are ignored. Therefore only the first plotting vector code (underlined) will be executed. On the other hand a byte of the following form:

XX 000 XXX

will be executed as expected. Therefore as long as the most significant code (the 2 bit code) in the byte is nonzero the 000 code will be acted upon as expected.

EXPLANATION OF PROGRAM LINES

LINE(S)	COMMENT
100	loads a machine code routine that logically ANDs bytes at memory locations 1060 and 1065
150-184	changes cursor color and allows cursor to flash
190-220	checks joystick movement
310-320	adds 3 bit plotting vector code to current byte
340-430	takes care of moving the cursor one position back and updating the bytes in the shape list
500-540	prints out the complete shape table on screen or printer (user's choice)

When drawing your shapes, you should plan your movements to minimize the number of moves you make. This will give a shorter shape table and a faster drawn shape. The program currently allows 14 shapes of 40 by 40 dots in a table. Since you probably won't want to fill in such a large area you can trade the size of each shape for the number of shapes in the table. This is done by modifying the 2 indeces in the array "m%". The fist index relates to the number of shapes in the table. The second index relates to the size of the shape. Realistically you can get away with a much smaller shape size thus allowing more shapes per shape table.

To reiterate, once you finish drawing, the numbers that this program provides you should be put into DATA statements within a graphics program of your own design. Thus this program's main utility is to serve as a design aid. It does not draw for you, but upon completing your drawing, will provide you with the necessary numbers that correspond to that drawing via shape table syntax

```
 90 DIM m%(13, 799)
100 FOR i = 0 TO 8: READ d%: POKE 1056+i, d%: NEXT
105 DATA   58,41,4,230,0,50,41,4,201
110 &  initialize editor
120 mul% = 1: ln = 0: st = 0: x = 0: y = 0: cl = 15: dir% = 1
125 GR: PRINT "use joystick to move cursor"
130 PRINT "fire buttons toggle cursor": PRINT "POSITION CURSOR TO START POINT"

135 PRINT "PRESS KEY ZERO WHEN FINISHED";
140 &  main loop
150 IF PDL(7) OR PDL(9) THEN  cl = 16-cl
160 IF st THEN  COLOR  = 0: GOTO 180
170 COLOR  = cl
180 PLOT x, y: FOR i = 1 TO 50: NEXT: IF st THEN  COLOR  = cl: GOTO 184
182 COLOR  = 0
184 PLOT x, y: FOR i = 1 TO 50: NEXT
190 p = PDL(5): IF p <> 1 THEN 200
192 IF dir% = 0 AND cl = 1 THEN  p = 0: GOTO 200
194 dir% = 0: y = y-1: IF y < 0 THEN  y = 0: p = 0
200 IF p = 4 THEN  dir% = 2: y = y+1: IF y > 39 THEN  y = 39: p = 0
210 IF p = 2 THEN  dir% = 1: x = x+1: IF x > 39 THEN  x = 39: p = 0
220 IF p = 8 THEN  dir% = 3: x = x-1: IF x < 0 THEN  x = 0: p = 0
230 pp = PDL(13): IF st = 0 OR pp <> 11 THEN 270
240 m%(sh, 799) = ln+1+(m%(sh, ln) <> 0): sh = sh+1
250 HOME: INPUT "End shape table? (Y/N): "; a$
255 IF a$ = "n" OR a$ = "N" THEN 120
260 GOTO 500
270 IF pp = 0 THEN  st = 1: VTAB 23: HTAB 1: PRINT "keypad # ends shape defini
tion": PRINT "keypad * erases last move   ";
280 IF st = 0 THEN 150
290 IF p <> 1 AND p <> 2 AND p <> 4 AND p <> 8 THEN 340
300 IF cl = 15 THEN  dir% = dir%+4
310 IF mul% = 64 AND (dir% = 0 OR dir% > 3) THEN  ln = ln+1: mul% = 1+(m%(sh,
ln-1) < 8)*7
320 m%(sh, ln) = m%(sh, ln)+dir%*mul%: mul% = mul%*8
325 IF mul% > 64 THEN  mul% = 1: ln = ln+1
330 GOTO 150
340 IF pp <> 10 THEN 150
350 mul% = mul%/8: IF mul% = 0 THEN  ln = ln-1: mul% = 64
355 IF ln < 0 THEN  ln = 0: mul% = 1: GOTO 150
360 IF mul% = 64 AND m%(sh, ln) < 64 THEN  mul% = 8
365 IF m%(sh, ln) < 8 THEN  mul% = 1
370 t% = mul%*(7*(mul% <> 64)+3*(mul% = 64))
375 POKE 1060, t%: POKE 1065, m%(sh, ln): CALL 1056: t% = PEEK(1065)
380 m%(sh, ln) = m%(sh, ln)-t%: t% = t%/mul%: oy = y: ox = x
390 IF t% = 0 OR t% = 4 THEN  y = y+1
400 IF t% = 1 OR t% = 5 THEN  x = x-1
410 IF t% = 2 OR t% = 6 THEN  y = y-1
420 IF t% = 3 OR t% = 7 THEN  x = x+1
430 COLOR  = 0: PLOT ox, oy: GOTO 150
500 TEXT: PRINT "output shape table data to printer? (Y/N): "
505 GET a$: IF a$ = "Y" OR a$ = "y" THEN  PR #1
510 PRINT sh; " 0 "; : sz = 2*sh+2: FOR i = 1 TO sh
515 v% = sz/256: PRINT sz-v%*256; " "; v%; " ";
520 sz = sz+m%(i-1, 799): NEXT: PRINT: PRINT
530 FOR i = 0 TO sh-1: FOR j = 0 TO m%(i, 799)-1
535 PRINT m%(i, j); " "; : NEXT: PRINT: PRINT: NEXT
540 PR #0
```

PRODUCT REVIEW: Troll's Tale - (1984) Sierra On-Line Inc.

Troll's Tale is not an ordinary text adventure game. It is an excellent example of what the development of these games with the addition of graphics can be like. Although the pattern of this game is quite typical, it has some interesting characteristics.

The objective is quite simple. King Mark's treasures are lost. The player must journey through caves, huts, a road, a beach, a house, a tree and other places in order to recover all of the 16 missing treasures. Messages and strange warnings seem to be clues leading to the solution of the game, but they are not. They do not affect the outcome but tend to only decorate the situation. Among some of the treasures are: a silver seashell and cup, a ring, a fiddle, a gold bar, etc. When all 16 treasures have been found, specific instructions must be followed to uncover the secret of the Troll's Tale. I have been told not to reveal it.

Throughout the game, the player is a spectator to a dazzling display of superb graphics. Each screen is instantly drawn taking full advantage of Adam's graphic capabilities. A lot of attention is placed on the visual aspect of the story. A short length fanfare is sounded after the recovery of each treasure and at the end. Another departure from the usual, is the use of the joystick to select options rather than enter commands with the keyboard. All possible options for each scene are listed at the bottom of the screen and thus a response such as "I cannot do that" is never seen. On the whole, this game is pleasant to look at and a breeze to play.

Because the only objective is to find the lost treasures, and since the options to each scene are presented before you, the element of trial and error overrides any trace of real challenge. At times, it's often tempting to simply cycle through the game rather than to think it out logically. After playing the game several times, it becomes tedious and only the fine graphics are appreciated. This is true for children as well since no real substance is found in the game. The conventional concept behind a good text game is lacking. There are no mysteries or puzzles to solve here. The only obstacle preventing you from retrieving a treasure is the troll, but it can easily be overcome.

Troll's Tale could have been improved with the addition of different skill levels and if being eliminated prematurely were possible. What this game lacks is the stimulus to use and sharpen one's deduction and thinking capabilities, the major focus of text adventure games. In general, Troll's Tale has some unique characteristics, but is somewhat on the borderline of being a challenging text game as we've come to expect them to be.

Troll's Tale is one of several software titles which were never released commercially by Coleco. What this means is that you can't go out and buy this game commercially. You can however have access to it from public domain sources such as users' groups, as it is now considered public domain software.

PRODUCT REVIEW: The Reedy Entertainment Pack 1

Developer: Reedy Software, 10085 - 60th St. S.E., Alto, MI 49302

Price: US $21.95 DP, $19.95 Disk

Entertainment Pack 1 is a collection of three graphic games. Included are Connect 4, Blockade, and Slide Puzzle. This package is self-booting. To initiate game play you simply insert the tape or disk in the drive and hit reset. A directory screen will then appear where you can select which of the three games you want to play. When a game is selected and fully loaded into memory, another menu screen is presented allowing you to select either a one-player game, a two-player game, the game instructions or the option to return to the game directory to select a different game. All three programs load very fast through binary files.

Connect 4 is a variation of the classic Tic-Tac-Toe strategy game. Colored blocks are used instead of 0's and X's while poles replace the familiar grid. This game is a lot more challenging because it offers an expanded grid in the form of 8 poles which can be stacked 8 blocks high. Players alternate stacking blocks on poles. The first to stack four of his colored blocks in a row either vertically, horizontally, or diagonally wins. Connect 4 is an interesting game to play especially with the excellent sound effects which have been incorporated.

Blockade is another game everyone is familiar with. It is none other than the ever popular breakout game. Slight variations have been made to make game play more appealing. Pinball sounds and effects are added to accomplish this. You use the hand controller to keep the bouncing ball striking against the wall to eliminate blocks and earn points. One or two player action is possible with individual skill levels. A pause feature is also included.

The third game, Slide Puzzle, is a scrambled number puzzle. There are 24 numbered pieces which you must slide back into the correct order. This game is great for very young children who are just starting to count. High resolution graphics and sprites are used exclusively. The visual presentation and the contents of this game are excellent.

All three games in this Entertainment Pack 1 make good use of sound and graphics, are easy to use and well written. With Connect 4 and Slide Puzzle, you have the choice of using either the hand controllers or the keyboard. One thing which I would have liked to see would have been the addition of a couple of other games. Without taking anything away from Slide Puzzle which I consider to be a professional program, most users are limited to the first two games as offering any real challenge. A game involving a higher level of strategy or even a text adventure game with graphics would have help make this package even more attractive and complete. Nevertheless, the three games offered are well done and anyone looking for more game programs will not be too disappointed.

```
 1 &
 2 &       Jason Rheindel - 1986
 3 &
10 TEXT: HOME: PRINT TAB(12); "BULLSEYE"
20 VTAB 5
30 PRINT "In this game, up to 20 players throw darts at a target"
40 PRINT "with 10,20,30, and 40 point    zones. The object is to get 200"
50 PRINT "points.": PRINT
60 PRINT "THROW  DESCRIP. PROBABLE SCORE"
65 PRINT "*******************************"
70 PRINT " 1     F-OVERARM BULLSEYE/MISS"
75 PRINT
80 PRINT " 2     S-OVERARM 10,20or30 point"
90 PRINT " 3      UNDERARM ANYTHING"
95 PRINT
100 DIM a$(20), s(20), w(10): m = 0: r = 0: FOR i = 1 TO 20: s(i) = 0: NEXT i
110 INPUT "How many players? "; n: PRINT
115 IF n < 2 THEN  PRINT "2 or more, please.": GOTO 110
116 IF n > 20 THEN  PRINT "Too many.": GOTO 110
120 FOR i = 1 TO n
130 PRINT "Name of player #"; i; : INPUT a$(i)
140 NEXT i
150 r = r+1: PRINT: INVERSE: PRINT "ROUND "; r: NORMAL
160 FOR i = 1 TO n
170 PRINT: PRINT a$(i); "'s throw"; : INPUT t
175 PRINT
180 IF t < 0 OR t > 3 THEN  PRINT "INPUT 1,2,or 3!": GOTO 170
190 ON t GOTO 200, 210, 200
200 p1 = .65: p2 = .55: p3 = .5: p4 = .5: GOTO 230
210 p1 = .99: p2 = .77: p3 = .43: p4 = .01: GOTO 230
220 p1 = .95: p2 = .75: p3 = .45: p4 = .05
230 u = RND(1)
240 IF u >= p1 THEN  PRINT "BULLSEYE!! 40 points!": b = 40: GOTO 290
250 IF u >= p2 THEN  PRINT "30-POINT zone!": b = 30: GOTO 290
260 IF u >= p3 THEN  PRINT "20-POINT zone": b = 20: GOTO 290
270 IF u >= p4 THEN  PRINT "WHEW! 10 points.": b = 10: GOTO 290
280 PRINT "MISSED the target! Too bad.": b = 0
290 s(i) = s(i)+b: PRINT "TOTAL SCORE ="; s(i): NEXT i
300 FOR i = 1 TO n
310 IF s(i) >= 200 THEN  m = m+1: w(m) = i
320 NEXT i
330 IF m = 0 THEN 150
331 FOR i = 1 TO 1000: NEXT i: HOME
335 PRINT "*********************************"
340 PRINT "We have a winner!!"
350 FOR i = 1 TO m: PRINT a$(w(i)); " scored "; s(w(i)); " points.": NEXT i
352 PRINT
355 PRINT "*********************************"
357 FOR i = 1 TO 20: PRINT CHR$(7); : NEXT i
360 PRINT: PRINT "Thanks for the game."
370 END
```

```
 2 &  3D Surface
 4 &  DAVID LILLY (1985)
 8 &
10 DIM g(43), p(27, 17)
20 HGR2: HCOLOR  = 3
30 FOR i = 1 TO 43: g(i) = 159-(6*(i-27))*(i > 27)
35 NEXT i
40 y = -8: FOR i = 127 TO 31 STEP -6
50 x = -13: FOR j = 128-i TO 284-i STEP 6
55 jc = (j+5)/6
60 z = SIN(.05*(x*x+y))*5
70 IF z+i < 0 THEN  z = -i
80 IF z+i <= g(jc) THEN 110
90 IF j = 128-i THEN  HPLOT j, g(jc): GOTO 130
100 HPLOT  TO j, g(jc): GOTO 130
110 g(jc) = z+i: IF j = 128-i THEN  HPLOT j, z+i
115 GOTO 130
120 HPLOT  TO j, z+i
130 p(x+14, y+9) = g(jc): x = x+1: NEXT j: y = y+1
135 NEXT i
140 FOR x = 1 TO 27: HPLOT 6*x-5, p(x, 1)
150 FOR y = 2 TO 17: HPLOT  TO 6*(x+y)-11, p(x, y)
160 NEXT y: NEXT x: FOR x = 1 TO 27
170 HPLOT 6*x-5, p(x, 1) TO 6*x-5, 159: NEXT x
180 FOR y = 2 TO 17
190 HPLOT 6*y+151, p(27, y) TO 6*y+151, 165-y*6
195 NEXT y
200 HPLOT 1, 159 TO 157, 159 TO 253, 63
```

```
 5 &   HEXAGON
 7 &
10 pi = 3.14159
20 c = COS(pi/3)
25 s = SIN(pi/3)
30 a = COS(pi/36)
35 b = SIN(pi/36): sf = .95
40 x = 95: y = 0: cx = 140
45 cy = 96: sc = 1.16
50 HGR2: HCOLOR  = 9
60 FOR j = 1 TO 40
70 FOR i = 0 TO 6
80 sx = x*sc+cx: sy = cy+y
90 IF i = 0 THEN  HPLOT sx, sy
100 HPLOT  TO sx, sy
110 xn = x*c-y*s
115 y = x*s+y*c: x = xn
120 NEXT i
130 xn = sf*(x*a-y*b)
135 y = sf*(x*b+y*a): x = xn
140 NEXT j
150 GET key$: TEXT
```

```
  5 HIMEM :51455
 10 GOSUB 1000
 20 HGR: HCOLOR  = 1
 30 ROT  = 0: SCALE  = 4
 40 FOR y = 40 TO 120 STEP 40
 50 FOR x = 50 TO 200 STEP 50
 70 DRAW 1 AT x, y
 80 NEXT x: NEXT y
 90 END
1000 POKE 51456, 1: POKE 51457, 0
1010 POKE 51458, 4: POKE 51459, 0
1020 m = 51460
1030 POKE 16766, 0: POKE 16767, 201
1040 READ a
1050 IF a = 8 THEN 2010
1060 READ b
1070 IF b = 8 THEN 2000
1080 x = b*8+a
1090 POKE m, x: m = m+1
1100 GOTO 1040
2000 POKE m, a: m = m+1
2010 POKE m, 0
2100 RETURN
2110 DATA  2,2,2,7,4,4,7,7,4,4,5,5,4,4,5
2120 DATA  5,6,6,5,5,6,6,7,7,6,6,7,8
```

A P E Software presents...

Electronic Game Pack

Now play exciting computer versions of the following popular games. Plus one new one especially created for this package. Games included are Battleship, Mastermind, Backgammon, Three Dimensional Tic-Tac-Toe, and Robot Miner. These are one player games that let you match wits with Adam. The games make extensive use of Adam's sprite, color graphic, & sound capabilities. All are fully documented. We even include a backup utility. To order your copy, send $29.95 plus $2.00 for postage (Que. Resid. add 9% sales tax)

A P E Software
4756 Lalonde
Pierrefonds, Quebec
H8Y 1V2

**********ADAM**********

ADAM EVALUATION CLUB

ORIGINAL
COLECO SOFTWARE & HARDWARE
LOWER PRICES!!!!

CP/M 2.2(DISK ONLY)...$55.00
-Upgrade your ADAM to the world of CP/M
ADANCALC(SPREADSHEET).$50.00
SMARTLOGO.....$45.00
SMARTFILER.....$40.00
SMARTLETTERS & FORMS..$40.00
COLECO HOME LIBRARY...$45.00
RECIPEFILER.....$35.00

2nd TAPE DRIVE.....$55.00
POWER SUPPLY(no case).$55.00
ADAM KEYBOARDS.....$40.00
ADAM PRINTER RIBBON...$15.00
WHITE HAND CONTROLLER.$15.00
DISK DRIVES.....$325.00
-PLEASE call for availability

-WE NOW HAVE A LIMITED # OF SUPERGAMES:ZAXXON (DP)..$35.00
DRAGON'S LAIR(DP).$35.00

CARTRIDGES AVAILABLE
phone for list and prices

Data Drive Speed Tester..$15.00
Make sure your drive is running at peak efficiency. Instructions for fine tuning provided.

*Public Domain Software Available
all CP/M 2.2 software available (WORDSTAR, DBASE II, COBOL..and many more all for your ADAM !!

NEW COLECO GAMES ON SONY TAPE inc -Jeopordy
-Family Feud
-Troll's Tale
-Pinup(Nude Graphic) 3.5 foot centerfold using your ADAM printer.

We now have a special on unlabelled Naschau disks
10 for $17.00 or 1 for $1.85

-AND DON"T FORGET OUR SOFTWARE EVALUATION CLUB. WE HAVE MOST GAMES & PROGRAMS ON DP OR DISK FOR YOU TO TRY AT $15.00 EACH.
-MEMBERSHIP IS $10.00.
-POSTAGE & HANDLING INCLUDED
-QUEBEC RESIDENTS ADD 9% TAX
-FOR OUR CATALOG & FURTHER INFO WRITE TO: ADAM EVALUATION CLUB
3811 PRUD'HOMME #14
MONTREAL, P.Q.
H4A 3H8
(514) 487-0110
9:00 to 4:00 ONLY