SYNTAX

2.6
JAN-FEB 87

TABLE OF CONTENTS

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

POSTAGE PAID / PORT PAYE
Permit
No. 7152
Montreal, Que.

This is the last issue of year 2 and that means it is time to resubscribe for the upcoming year. Before I try and convince you that resubscibing to Syntax would be an excellent idea, let me first go through a few "technical" details. Syntax is published every 2 months starting with the March-April issue. Each issue is numbered from X.1 to X.6 inclusive, where X stands for the year. By year we mean subscription year March to the following Febuary, not January to December. Thus our next issue will be 3.1 because it is the first issue of our third year. When you join or rejoin Syntax you are expected to get all the issues for that (subscription) year. If you join anytime after March we backdate your subscription to start from X.1. We do this for several reasons. Book keeping is a lot easier for us if we know where every member is in regard to their subscription. But more importantly, we seldom (if ever) repeat material and often like to build on earlier material, so having a full year allows you to grow with us. It also allows us to know about what level our audience is at. So it would probably be a good idea to pick up the first year if you don't have it too. Very little information is dated, maybe a page or two of adds each issue, if that many, is all.

Now about FCAUG itself. A lot has been changing for the group over the year. Most of the staff started as students or people with a lot of time on their hands. Unfortunately, many have either moved away or got jobs or got involved in the production and programming end of things or got married or a lot of other things that are certainly great for them, but not so good for the magazine. So it has been getting harder to get Syntax out. We need help and we need encouragement.

Fortunately, we are getting some new blood and this is exactly what we need because I for one want to keep this thing going. Syntax provides a national link for Adam users and without it all the isolated members, even in the large centers like Montreal and Toronto, would eventually dry up. Now members can share information, programs and ideas with people all across the country. We provide a forum for vendors to sell their products and users to buy new things. And we do it in Canada, at Canadian prices. (Often this involves our buying in bulk straight from American companies so that you don't have to worry about the customs hassles. Or we try and support good, local Canadian developpers reach the Canadian market.)

We hope that you subscribers out there will also become contributers soon. FCAUG is a club and you can and should make your ideas and questions known through the magazine, is one way to look at your Syntax subscription. But FCAUG is also people who write the articles and ask the questions and speaking as a person, as much as I love to explore the world of Adam, I still need to work, eat, etc. What I need to know then is what do <u>you</u> want Syntax to be? An experts' corner, a trading post, whatever you want it to be, you have to help make it. So please participate. If you can't contribute, then please don't work against us and read Syntax for free, subscribe.

In the next year we will have technical manuals, software from Europe and a lot more developments in store for you. If you're thinking about rejoining, now is certainly the time to do so.

L E T T E R S

Dear FCAUG,
I received my Nov - Dec issue of SYNTAX and have read it from cover to cover thoroughly enjoying all of your articles. I would be sorry to see you reduce the number of issues you publish. SYNTAX to me is the most important part of the users' group. I typed in your program directory modifier (SYNTAX 2.5) and after correcting my typing errors, the program ran fine for my disk drive and tape drive #1, but it would not recognize my second tape drive. I changed "18" in lines 150 and 160 to "24" and the program now recognizes my second tape drive. I am glad to see you are expanding your product list.

W. Stewart
Burlington, Ont.

Our apologies for the error in the program. We put in the hex value (18) instead of the required decimal value of 24. Thanks for pointing it out.

Dear FCAUG,
In SYNTAX 2.1, you mentioned the computer's clock and suggested a way to use it as a randomize routine. I would like to know if there is a way to use it to create a clock that could hold the time of the day. Keep up the good work.

Daniel Gouin
Ste-Foy, Que.

The TIMEX program by Chris Hills does precisely what you are looking for. The advantage of this program is that it remains active in SmartBASIC even when you work on other programs.

Dear FCAUG,
I have been in contact with Data Ribbon Ltd. to obtain new printer ribbons as mentioned in SYNTAX 2.4. The company advises me that the cost has increased from $7.00 to $12.00 for new cartridges. They will still honor the cost of $4.00 to refill used cartridges. I have also found a supplier of pre-formatted blank tapes at exceptional prices: 1-10 @ $1.95 or 10 and up @ $1.75 in American funds. The company, M.C.P. Software, P.O. Box 64, Marlton, NJ, 08053 also has a 4 page leaflet on various home management and educational programs as well as fanfold stationary products. I have checked with Canada Customs and you are allowed $40.00 Canadian funds into the country duty free. I feel that it is not worth it to buy an audio tape and make your own at this cost. Finally, I found an excellent book on Adam entitled, "The Coleco Adam Starter Book", written by Christopher and Jonathon A. Titus. In my opinion, this is the best formatted book that I have found. It is ideal for teaching as it is designed like an old fashioned school book. Every chapter is broken down into experiments with review questions at the end of the lesson. The solutions to the questions are contained at the end of the book. This book has an excellent section on sound, better than Adam's

Companion in my opinion. Perhaps you could write a review for other members at a later date.

Garry Lording
Gananoque, Ont.

Thanks for the update on the new prices for new printer ribbons from Data Ribbon Ltd. On a price to performance basis, I have to disagree with you on ordering formatted tapes from the U.S. Not only do you have to pay each order in US funds (plus the $5.00 minimum shipping charge) but you have to go through Canadian customs which can involve considerable delay and inconvenience. In the end, what you will get is a cheap audio tape which is formatted but hardly acceptable to any serious computer user. If you format you own, you can at least choose a brand name tape (Sony, TDK, Maxell, etc.) of superior quality. If you shop around and buy these 60 min. tapes in quantities of 10 or more, you can usually get them for $2.00 or less.

Dear Sirs,
You may want to pass along this small note to other members in the Ottawa-Gatineau region. Loran digital data packs for the Adam are available at Canadian Tire, 2010 Ogilvie Road, Ottawa, Ont. Tel. (613) 748-0637, under product #69-3024-4 at a price of $7.99 each. This particular store has indicated that they will keep this item in stock as long as there is a demand for it.

Jean-Marc Roy
Touraine, Que.

Dear Sirs,
I have one question which has really been puzzling me. In the super game pack version of "BC II, Grog's Revenge" how come everytime I make it to the "Meaning of Life" level and I go into a cave, the entire game completely crashes? (I mean black with no way to regain control without pulling the reset switch.) I tried exchanging it at Canadian Tire for another copy thinking that I had a defective one. It happened with the new one also and the same problem occurrs on my friend's game. Any suggestions as to why I am encountering this problem?

Daryl C. Heinsohn
Dowling, Ont.

Sorry, Daryl we do not have an explanation for you. If anyone does, we would like to hear from them regarding this quirk.

SYNTAX is published bimonthly by First Canadian Adam Users' Group (FCAUG), P.O. Box 547, Victoria Station, Westmount, Que., H3Z 2Y6 Subscription Rates: 6 issues for $20.00 CDN. Second Class Mail Registration No. 7152. POSTMASTER: Send all address changes and notice of undelivered copies to the above address. Return Postage Guaranteed.

An Open Letter To Members of FCAUG

Two issues that may well be of interest to many of you, have been on my mind for some time, and I'd like to know if the group can do anything about them. The first concerns the supply of replacement parts for the Adam. From time to time, I have seen classified advertisements in both the Montreal Gazette and the Ottawa Citizen in which Adam systems have been offered for sale. What do you think of the idea of having the group purchase such systems to make their parts available to members who may need them? Would you mind having your membership fee increased by a few dollars to cover the costs of the initial purchase? Sale of the parts to members could result in cost recovery. The second issue is about computer knowledge. Home computer owners may be divided into three broad categories:

(a) those who use the computer only to play games, as a word processor, or with commercially-available software.
(b) those with good computer knowledge who can make Adam do many things not mentioned in the manuals or in the various books that have appeared on the subject; and
(c) a large group in between who would like to learn some of the tricks used by the preceeding group but who lack the knowledge and understanding.

What I am suggesting is that one of the more knowledgeable among the members write a series of articles explaining things starting from basic principles. Much of such information already appears in SYNTAX, but most of it is at a level where many of us cannot understand. Some feedback on these suggestions would be appreciated.

Sydney Abbey
Ottawa, Ont.

There is no need to increase the fee to provide the service you are suggesting. Not only have we reduced the fee for this year to but we are already in possession of many spare parts for which are available for sale to any member who is . We do agree with you that it is a good idea to ie units of individuals who have decided to to hear from

I have disassembled (by hand) the TEXT routine (for SmartBasic 2.0) and have come up with the following equivelent pokes.

V1.0	V2.0	Function	Def
17059	17184	Border Color	0
17115	17240	Text Color	240
17126	17251	Flash Color	15
17164	17289	Primary Screen fill	32
17175	17300	Secondary Screen fill	32
17198	17322	Bottom Margin	23
17199	17323	Right Margin	30
17201	17325	Top Margin	0
17202	17326	Left Margin	1

The TEXT routine (17046-17225) is now at (17171-17349) in SmartBasic 2.0 and is one byte shorter. Equivelent areas loaded by the TEXT routine areas follows.

V1.0	V2.0	Description	Value for TEXT Mode
17009	16777	Print Character indicator	255
17011	16948	Frequency of flashing	12
17012	17163	VRAM address of Name Table for flashing	6144

The TEXT routine also calls the INIT screen routine at (17334-17388) and in V2.0 at (17502-?). I have just started to disassemble this routine and as the V2.0 routine is shorter than V1.0, I have yet to determine the end of that routine. Because of the added features of V2.0, I want to keep exploring it in order to render it useful to myself and others.

Guy-R. La Forest
St. Jacques, NB

SUBSCRIPTION RENEWAL FORM
Fill out and return to

CONVERTING WORDSTAR DOCUMENTS TO SMARTWRITER FILES
(and learning a bit more about the Adam in the process)

Back in issue 2.1 (p.14) we discussed the problem of printing WordStar saved files under CP/M for people without WordStar. When using the TYPE command you may get inverse characters and other bizarre screen effects when attempting to print one of these files. You may remember the solution - PIP the file with the Z option on, thereby stripping out all the WordStar control codes. But what if you want to take a WordStar file and use the built in SmartWriter program to print it out? WordStar is very popular on other CP/M machines as well as the IBM and compatibles so even if you don't have WordStar you may come across a text file produced by WordStar on a BBS or as a .doc file on CP/M. So knowing how to change it to a SmartWriter file could come in handy.

Before we look at specifics let's look at some general concepts. The Adam can use 2 operating systems - the built in EOS and the tape or disk based CP/M. It is the job of both of these O/S's to coordinate the computer's hardware resources in an effecient manner and create an environment that the programmer can make efficient use of. Both O/S's are a collections of software routines, written with this in mind. Both O/S's have programs written to work in their respective environments. This also means that an EOS disk, direcrory, and files are going to be different and therefore incompatable with CP/M. But when converting from WordStar to SmartWriter files, we are not only faced with a conversion from one type of O/S to another but a conversion from one program to another. There will be differences due to both these factors.

One difference between WordStar and SmartWriter files is in how formatting control codes are stored in the files. WordStar is more versatile than SmartWriter in that one document file can have several page widths and line spacings (among other things) set, whereas although SmartWriter files can change their standard settings, once changed they are set for the whole file. WordStar attains its flexability by inserting control codes in the text that the WordStar program can understand and act upon as it prints through the file. Other kinds of information are stored by setting the high bit of a character byte. This philosophy takes advantage of the fact that character information in the computer is represented via the 7 bit ASCII code. Knowing this, the designers of WordSTar used the unused high bit to convey control information. Compare this with SmartWriter produced files. All control information is stored in the first 258 bytes of the file. It should be clear to you now that this 258 byte header can not store more than one format per file. SmartWriter files also make use of the unused high bit of the ASCII code but in a different way than WordStar. If this bit is set, the character is highlighted when displayed on the screen (red underline). Let's apply this information practically now and see what would happen if you downloaded a WordStar file using AdamLink II. First off the file will be an EOS file so that problem is solved. Second the file will have no header that SmartWriter can read. Third is that the file while be peppered with incomprehensible WordStar control codes.

The first problem to clear up, then, is how to get a proper SmartWriter header at the beginning of your WordStar file. Happily, the answer to this problem also answers a previously unsolved problem. You may remember the AdamLink update in Syntax 1.2 (p.13) where we talked about the program's inability to provide us with editable text files. We can now explain why. AdamLink II was designed to transfer text files from one computer to another. Since SmartWriter headers are specific to Adam systems and would only confuse other computers they are neither sent or received with files transferred with this program. Unfortunately for Adam users the transferred files, even if Adam to Adam transfers, become "A" type files and have no headers. And without the headers no editting changes can be resaved with the file. So the two problems are the same - how do we add a SmartWriter header to the beginning of a text file. It's very simple. Load the file into memory. Now insert a <RETURN> at the very beginning of the file. Making sure that this is the only thing that you can see on the screen, clear the screen. Now store the file. All of your changes will remain intact. For some reason a header is created and your file is stored away properly as an "H" file.

Stripping out the embedded WordStar control codes is also very easy. Since the set high bits will show up as red underlined (highlighted) letters all you have to do is to erase the highlighting of any text that you may come across on the screen. This will clean up the file but if your file is large be warned that this processs may take some time as you can only remove the highlighting (erase highlight) one screen at a time.In the end you will have a perfectly compatable SmartWriter text file.

If someone gives you a disk with a WordStar file on it (or you download a WordStar .doc file from a BBS with A CP/M modem program) and only have SmartWriter to print it out there are a few things you'll have to do. The first thing is to strip out the control characters with PIP.COM (Syntax 2.1). Then you must add a control character (^Z or 1A$) at the end of the file. Most CP/M text or word processors end the file with a ^Z, but WordStar is an exception. The ^Z must be added because later we will have to use a program that can only work properly with this CP/M convention in place. The DDT.COM program is the recommended tool for the job here. Since your files will be of different lengths you will have to find the last byte + 1 and I can't tell you which byte that will be. But DDT will tell you under the NEXT label as soon as you load the file in. So use this program and change the last byte +1 to 1A (hex). After you've done that all you have to worry about is transferring the file to an EOS file on an EOS disk. This is very easy with the CP/M CPMADAM.COM program, but there are some red herrings to watch out for as you go along. The CPMADAM program asks what kind of file the newly created EOS file should be (A or H). Always say "A". If you say H the transfer will take place properly but no Adam program will be able to read it due to the rules of EOS file format conventions. If you say "H" Adam will look for either a SmartWriter or Basic binary header. Finding neither you will get a "can't read from this file" message in Basic (errnum = 5) or "can't access this file" message SmartWriter.

Assuming you've stuck with me so far, your file should now be on an

EOS disk as an "A" type file. We are now in the same position that we were if we had used AdamLink II to download the file. So we could put a header on our file now, but let's wait, there's still one more thing to do. Remember the ^Z we inserted? That is only one of the conventions that CP/M text editors use to process their text files. Pi symbols ($0A - line feed) and <TAB> symbols also have to be either stripped or converted. Fortunately, we can let a Basic program do all the work for us. We found just such in the American magazine ECN. It is called convert. To use it just type in the listing that you see later in this issue and run it. The program will provide you with all the necessary prompts. This program performs the same function that the CONVERT.COM program does in the CP/M manual, except it does it in the opposite direction. Thus it is an indispensible tool for anyone that converts files from CP/M to EOS formats. After running this program you can now add the header to your file. And after all that you are finished.

```
 10 LOMEM :37550
 15 base = 27500
 20 PRINT "CP/M to Adam fix program"
 30 PRINT "This program fixes files"
 40 PRINT "transfered to Adam from CP/M"
 70 PRINT "Enter name of drive"
 80 INPUT " Drive (d1,d5,d6)"; drive$
 90 PRINT CHR$(4); "catalog,"; drive$
120 PRINT "Enter name of file just loaded"
130 PRINT "from CP/M"
140 INPUT " Filename: "; sfile$
150 PRINT "Enter name of new file"
160 INPUT " Filename: "; dfile$
170 maxln = 250
180 length = 10000
190 lastchr = 0
200 done = 0
210 bn = 0
220 GOSUB 2000
230 GOSUB 3000
235 PRINT "Block "; bn; " complete"
240 IF done <> 1 THEN 220
250 PRINT
260 PRINT "Conversion program is complete"
270 END
2000 REM    read text block
2005 ONERR  GOTO 2150
2010 adr = 0
2020 HTAB 1: PRINT CHR$(4); "open "; sfile$
2030 HTAB 1: PRINT CHR$(4); "read "; sfile$
2040 GOSUB 7000
2050 GET a$
2060 IF a$ = CHR$(26) THEN  done = 1
2070 POKE adr+base, ASC(a$)
2080 adr = adr+1
2085 IF adr < length-maxln THEN 2090
2086 IF a$ = CHR$(13) THEN 2110
2090 IF done = 1 THEN 2110
2100 IF adr <> length THEN 2050
2110 bn = bn+1
2120 lastchr = lastchr+adr
2125 blk = adr
2130 HTAB 1: PRINT CHR$(4); "close "; sfile$
2140 RETURN
2150 PRINT
2155 HTAB 1: PRINT CHR$(4); "close "; sfile$
2160 PRINT "Error occured accessing"
2170 PRINT " source file "; sfile$
2180 CLRERR
2190 END
 3000 REM    Write block routine
3005 ONERR  GOTO 3200
3010 IF bn <> 1 THEN 3050
3020 HTAB 1: PRINT CHR$(4); "open "; dfile$
3030 HTAB 1: PRINT CHR$(4); "write "; dfile$
3040 GOTO 3060
3050 HTAB 1: PRINT CHR$(4); "append "; dfile$
3060 adr = 0
3070 a = PEEK(adr+base)
3080 IF a = 10 THEN  count = 0: GOTO 3140
3090 IF a = 9 THEN  GOSUB 6000: GOTO 3140
3100 IF a = 26 THEN 3160
3110 IF a = 13 THEN  PRINT: count = 0: GOTO 3140
3120 PRINT CHR$(a);
3130 count = count+1
3140 adr = adr+1
3150 IF adr <> blk THEN 3070
3160 HTAB 1: PRINT CHR$(4); "close "; dfile$
3170 RETURN
3200 PRINT
3210 PRINT: PRINT CHR$(4); "close "; dfile$
3220 PRINT "Error occured accessing"
3230 PRINT " destination file"; dfile$
3240 CLRERR
3250 END
6000 PRINT " ";
6010 count = count+1
6020 IF count <> 8*INT(count/8) THEN 6000
6030 RETURN
7000 REM    dump data block routine
7010 FOR db = lastchr TO 0 STEP -1
7020 GET a$
7030 NEXT db
7040 RETURN
```

Machine Language Primer

We have looked at a few EOS system routines in the past, so we know how to use them. The problem lies in what's available to use. In this article, I will give you a short description of the most useful EOS routines.

In order to make the descriptions short I have used abbreviations, acronyms, and generally assumed you probably know more information about the ADAM than you think. Therefore, here is some filler information you will need before you can fully understand and make use of the EOS system routines list.

The first value in each system routine description is the hexadecimal address you would call to use the routine. The 2 decimal values in parentheses are the calling address broken into its lower and higher bytes, respectively. These are the 2 bytes you would place in a BASIC DATA list when poking in a machine language routine which calls the system routine. The capitalized words gives the routine name followed by extra comments in lower case. The IN: refers to what the routine expects in certain registers before the routine is called. The OUT: refers to information the routine places in registers when finished.

The registers used are shown in the IN:/OUT: sections as: A (8 bit accumulator), BC, DE, HL (16 bit register pairs), and IY (16 bit index register). The Z refers to the Zero bit being set to 1 in the flag (F) register while NZ refers to this bit being 0. You check this bit with a test/jump instruction (ie. JP Z,BITSET (jump to BITSET if Zero Flag bit is set to 1)).

Many of the routines transfer bytes from one place to another. Buffers are areas of memory that hold or will hold the transfer bytes. Therefore there can be source (src) or destination (dst) buffers. The routines will want the addresses where these buffers start. Buffers in Video Display Processor (VDP) Video memory (VRAM) are referred to as VRAM buffers, if in Z80 memory then just plain "buffers".

Block devices are the 4 tape and 2 disk drives. The devices are given device numbers to refer to them. The device numbers are 8, 24, 9, and 25 for the 1st, 2nd, 3rd, and 4th tape drives and 4 and 5 for the first and second disk drives, respectively. A block consists of 1024 bytes, the first block being block 0.

The following information is needed to use the VDP table routines.

TABLE #	TABLE NAME	ENTRY SIZE	STORAGE ADDRESS
0	sprite attribute	4 bytes	64868
1	sprite pattern	8 bytes	64870
2	screen (or name)	1 byte	64872
3	pattern	8 bytes	64874
4	color	1/8* bytes	64876

* 8 byte color table entry when in the hires graphic modes.

EOS System Routines List

- FC30 (48,252) COLD RESTART like pulling the ADAM reset switch.
IN: none OUT: none

- FCE7 (231,252) JUMP TO WORD PROCESSOR. IN: none OUT: none

- FC69 (105,252) READ 1 BLOCK from a block device.
IN: A=device #, BC=0, DE=block #, HL=dst buff OUT: none

- FCF6 (246,252) WRITE 1 BLOCK to a block device.
IN: A=device #, BC=0, DE=block #, HL=src buff OUT: none

- FC6C (108,252) READ KEYBOARD waits for a key. IN: none
OUT: Z=ok, A=key read or NZ=error, A=error code

- FD17 (23,253) LOAD ASCII PATTERNS FROM ROM TO VRAM.
IN: HL=starting ASCII code, BC=# of ASCII patterns to load,
DE=VRAM dst buff OUT: none

- FD1A (26,253) VWRITE write bytes from buffer to VRAM.
IN: BC=# of bytes, DE=VRAM dst buff, HL=src buff OUT: none

- FD1D (29,253) VREAD read bytes from VRAM to buffer.
IN: BC=# of bytes, DE=VRAM src buff, HL=dst buff OUT: none

- FD20 (32,253) LOAD VDP REGISTER. IN: B=VDP register (0-7)
C=data to load into VDP register OUT: none

- FD23 (35,253) GET VDP STATUS REGISTER which contains sprite
collision bit. IN: none OUT: A=VDP status register

- FD26 (38,253) FILL VRAM with a byte value.
IN: A=byte value to fill with, DE=# of bytes to fill,
HL=VRAM dst buff OUT: none

- FD29 (41,253) INITIALIZE VDP TABLE maps a VRAM address to a VDP
table #, table used by some routines, copy of VRAM table
addresses stored in memory for accessing convenience.
IN: A=table #, HL=VRAM VDP table address OUT: none

- FD2C (44,253) FILL VRAM VDP TABLE with entries from memory
buffer. IN: A=table #, DE=starting entry # (0=1st, 1=2nd, etc.),
HL=src buff, IY=# of entries OUT: none

- FD2F (47,253) READ VRAM VDP TABLE entries to buffer.
IN: A=table #,DE=starting table entry #, HL=dst buff,
IY=# of entries to read OUT: none

- FD38 (56,256) LOAD ASCII PATTERNS TO VRAM VDP PATTERN TABLE,
uses VRAM VDP table, loads ASCII characters 0 to 127.
IN: none OUT: none

MACROS

Lately there have appeared a number of modifications to the Coleco SmartBASIC interpreter. The modification or addition of most interest to myself is the ability to add Macro commands to SmartBASIC. The routine which allows the addition of Macro commands to SmartBASIC is published in Peter and Ben Hinkle's THE HACKER'S GUIDE TO ADAM, vol. 2 along with a Basic HELLO type program which installs the Macro commands on booting SmartBASIC. After reading the HACKER'S GUIDE (an excellent booklet that I would recommend without hesitation to any ADAM programmer) and analysing the machine code which executes the Macro commands I decided to embed the Macro routine in the SmartBASIC interpreter. This results in the original Lomem address setting of SmartBASIC not being disturbed resulting in the same amount of available free RAM space for users programs. Additionally upon booting SmartBASIC the macro commands will be active from the start without having to execute a HELLO program.

With the installation of Macro commands to SmartBASIC the user is now able to perform various operations at the stroke of a single key. As an example a Macro command can be installed to make the 'CLEAR' key active. In other words pressing the 'CLEAR' key on the keyboard will perform the same function as a CONTROL L operation which is to clear the screen. Likewise the 'DELETE' and 'INSERT' keys can be activated to delete a space (same as CONTROL O) and insert a space (same as CONTROL N). These are standard additions which enhance SmartBASIC however the real value in Macro commands becomes evident when Smart Keys I to VI are programmed to perform Macro instruction sets. For example I have installed the Macro command CATALOG <CR> for Smart Key I. Thus whenever Smart Key I is pressed the instruction CATALOG is printed on the screen and automatically executed. Examples of some useful Macro instructions are as follows:

Smart Key	Macro Command	Execution
CLEAR	CONTROL L (^L)	Clears screen
INSERT	^N	Insert 1 space
DELETE	^O	Delete 1 space
I	CATALOG <CR>	Gets Catalog
II	RUN <CR>	Runs Program
III	LOAD <CR>	Loads Program
IV	BRUN <CR>	Runs an 'H' File

Hence a typical work session on the ADAM computer begins for myself by first booting SmartBASIC. I then press Smart Key I to display the catalog. After deciding which file to run the arrow keys are used to step the cursor up the extreme left side of the screen until it resides on the same line as the program which is to be run. Smart Key II is then pressed to load and begin program execution. Users who do a fair amount of programming on the ADAM will appreciate the value of the above Macro commands.

This article will attempt to present a very limited explanation of the startup routines which the SmartBASIC interpreter executes along with detailed instructions to let the average programmer

duplicate my efforts and embed the Macro routine along with some Basic commands into the SmartBASIC interpreter. To implement this will require the programmer have and be capable of using programming tools such as a 'copy routine' along with an 'tape editor routine'. For the copy routine I would recommend QUICKOPY by GJMG ENTERPRISES as this routine is an excellent and reliable routine. Secondly for the editor routine I wrote my own but any routine which will read a specified block of data from tape/disk, allow it to be changed by the user and then re-written back to tape/disk would be adaquate. I understand SYNTAX now offers an editor routine called E-Z EDIT which should be more than adaquate. My definition of a good editor is one that allows the user to make multiple changes to a block (1024 bytes) of data in either hex or decimal format and then write the entire block back to the storage device.

Now lets begin. I will list all the necessary steps to embed the Macro routine into the interpreter in the following format:

STEP #n: Instructions

However first we must familiarize ourselves with some of the details on how to calculate the mapping of the address locations in RAM memory to the byte locations on tape/disk. Murphies law states that there is no possible way that RAM address 1000 would be stored as the 1000 th byte of the first block on the tape/disk.

In order to calculate and understand the RAM address to the respective tape/disk byte location mapping we require to understand how SmartBASIC is booted from tape/disk. First when the ADAM reset switch is pulled and there is a data pack or disk in one of the drives ADAM will automatically load block 0 from the tape/disk to RAM memory starting at address 52300 and up, for 1024 bytes. Upon completing this load of 1024 bytes ADAM then jumps to address 52300 and begins the execution of this machine code just loaded. Hence block 0 of the tape/disk is refered to as the 'boot block' and is responsible then for loading the SmartBASIC machine code into RAM memory. We then disassemble the boot routine to find out how it operates and the following is determined. The routine after performing various checks and initializations will read block 1 of the tape/disk and scan it for the entry BASICPGM2. When the entry is located the start block is determined and the length in blocks of the data is also determined from the BASICPGM2 directory entry. The booting routine subsequently sets the start address in RAM which becomes the address of the first byte which is loaded into RAM from the first block (start block) of BASICPGM2. Data is then loaded continiously from this address in RAM for the length of bytes as determined from the BASICPGM2 directory entry. Anyway the information that interests us is the start address and it's easy to determine that it is RAM memory address 256. In other words the boot routine, which is executing in the boot block area of RAM (52300 - 53323), starts loading the SmartBASIC interpreter starting at RAM address 256 and continiously for 28 x 1024 bytes. Secondly and just as important is the finding that after the boot routine completes execution it passes control (jumps to) to RAM address 256.

At this point the SmartBASIC interpreter has been loaded into RAM memory and is beginning to execute starting at address 256. Before we proceed to the next step of execution we can calculate the byte mapping between RAM memory and tape/disk from the above information. It's easy to see that the mapping is a simple 256 byte offset. Also it's known that the SmartBASIC interpreter data is stored in blocks 2 to 30 on the tape/disk hence the following formula can be used to find the block location BLK of RAM address RAM(adrs) on the tape/disk storage media.

```
BLK = 2 + (int[{RAM(adrs) - 256} / 1024])
```

Hence we now know which block on tape/disk corresponds to the RAM address, however to use the 'editor' routines we must calculate the location BYTE(loc) of this RAM address in the block of data (1024 bytes) can be found in. The second formula to find the byte location BYTE(loc) becomes:

```
BYTE(loc) = [(RAM(adrs) - 256) - ({BLK - 2} x 1024)]
```

Therefore RAM address 256 can be calculated to be in block:

```
BLK= 2 + (int[{256 - 256} / 1024 ])

BLK = 2 + (int[0]) = 2
```

And its location in block 2 is:

```
BYTE(loc) = (256 - 256) - ({ 2 - 2 } x 1024 )
          = 0 - 0
```

Hence byte 256 is byte zero (first byte) in block 2 of the tape disk.

As a second example and becouse we will use the information later lets calculate the tape/disk byte location of RAM addresses 12197 and 12198.

```
BLK = 2 + (int[{12197 - 256} / 1024 ])
    = 2 + (int[ 11941 / 1024 ]
    = 2 + (int[ 11.66 ])
    = 2 + 11
    = 13

BYTE(loc) = (12197 - 256) - ((13 - 2) x 1024)
          = 11941 - ( 11 x 1024 )
          = 11941 - 11264
          = 677
```

Hence RAM address 12197 is the 677th byte in block 13 of the SmartBASIC tape/disk. It then follows that RAM address 12198 is the 678th byte of block 13. We will later change the contents of these 2 bytes on the tape/disk.

ADAM CP/M 2.2 and the Video Display Processor

This describes what I have been able to find out about the video display mode used for ADAM CP/M. If you don't have the Data Manual for the TMS9918 family of Video Display Processors, I strongly advise you to pick up a copy. The TI publication number is MP010A, and my copy cost about $20 Canadian.

The BIOS cold boot segment initializes the VDP in Mode 1, which is Graphics II mode. The actual subroutine is at DE24. Graphics II mode treats the screen as three separate areas, each consisting of 8 lines of 32 columns. The bottom area is set up for the notorious SmartKey display panels, whilst the other two are set up 'normally'. Each area has its own Pattern Name Table (PNT), Pattern Generator Table (PGT) and Pattern Color Table (PCT). Sprites are also available (the cursor is a sprite).

The VRAM table origins are set up like this:

```
Pattern Generator Table: 0000  (6144 bytes)
Pattern Name Table:      1800  ( 768 bytes)
Color Table:             2000  (6144 bytes)
Sprite Pattern Table:    3800  ( 128 bytes)
Sprite Attribute Table:  3880
```

The BIOS subroutine at DE24 does the following:

Calls DEAA to set up VDP registers. The register values follow the call, in reverse order of register number (ie: register 7 first). The values used for setup are:

```
#7   07 (cyan background ... modifiable by CONFIG.COM)
#6   07 (Sprite PG table  at 3800)
#5   71 (Sprite Attribute at 3800)
#4   03 (Pattern Table    at 0000)
#3   FF (Color Table      at 2000)
#2   06 (Pattern Names    at 1800)
#1   80 (Mode 1, 16k VRAM)
#0   02
```

Calls DEB8 to fill the Color Table. Then copies the character set patterns from boot ROM location 0102 to VRAM. The sprite data for the cursor is copied from DC93 (cursor pattern) and DC8E (sprite attributes). The cursor pattern and color can be modified by CONFIG.

The Pattern Name Table is filled with blanks, and the screen is cleared. Subroutine DF3D is called to write VDP data for the SmartKey panels, and EDB4 displays them (if enabled). The patterns for the 'ADAM CP/M' banner screen are then written out, the colors set and we are done.

Book Review: The Hacker's Guide to Adam Volume 2

This volume should really be called the Adam User's Guide to SmartBasic, but we won't quibble because the Hinkles have come up with another winner. Although not as extensive in its overview of the Adam system as its sister companion (see the review of volume 1 in 2.5), this volume is a very important addition to the Adam users programming arsenal in that it concentrates on one thing - demystifying the main programming tool of the Adam system, the SmartBasic interperater.

As I said in my review of volume 1, this Hinkle publication will not appeal to every one of you. But let's look at who it will appeal to to see if you might be interested. The first candidate would be your average Basic programmer. If you've done a lot of Basic programing you very quickly start to run into a few walls with SmartBasic. We have explained some of the more common problems and how to get about them. Things like sound, video, and sprite programming would be included in this category. The first volume covered these extraneous but useful routines as well. What the second volume does is to concentrate on the routines that are already part of Basic as it is. For example we all know how to change the screen color with a simple POKE statement. What this book does is show you that this simple POKE statemnt is actually part of a machine language routine that, when called, is executed and performs a screen color change. That's why you have to say TEXT after you POKE at 17115; poking by itself does nothing, you have to make the change and then execute the proper routine to see its effect. In this case, nothing more complicated than typing TEXT. What a novice programmer that has stretched Basic to its limits can learn then is the underlying concepts of how Basic works. Even better the concepts learned here are universal, so you really are learning about how all high level languages work.

So now you know that our nice user friendly SmartBasic is really just a collection of subroutines. Knowing this, you can now explore the different types of subroutines that comprise or more properly is SmartBasic. The Guide tells you where the routines are and how they work so that you can then start changing them to your needs. This benefits machine language programmers because they don't have to search around to see where Coleco put their specific routines and educates the Basic programmer looking for new horizons.

It goes without saying that this kind of information can improve your programs. For example knowing how the computer defines the screen boundaries may not seem that interesting. However, if you know how Adam normally does this you can make Adam do things like windowing. This allows you to change the size of the screen, move the screen about, overlap the screen with a completely different screen, and then switch back between screens. We hope to have an example of this in Syntax shortly.

This book looks at every routine in Basic, every command and every data table. Nothing is hidden because every single byte is accounted for. This does not mean that each byte of each routine is shown. If you want that you would have to disassemble each routine and the

listing, never mind the explanation, would be huge. What the Hinkles do is tell you where the routine is what it does and what it needs to do it. This is all you need to know. But if you do want more you can disassemble a particular routine because they tell you where it is and how long it is.

If you want to know how Basic processes your commands in immediate and programming modes the Hinkles will show you how. This involves things like tokens, crunch codes, the process called parsing, and the formation of program tables. As you type in a line or as a line is being interperated when it is being loaded in from a program, SmartBasic is breaking up the line and storing its component parts away in specific locations in memory. The line numbers, string variables, string values, commands, integer values, real number values, and number variables are all stored differently and in different areas of the memory. This makes it effecient for Basic to execute the program but darn near impossible for us humans to understand. The Hinkles provide 2 short programs that show how a number is stored in memory and how a line gets broken up into crunch code in memory. The Hinkles might have gone further here. Knowing how the code is efficiently stored allows you save your program in this form on tape. Your program can then take up less space on the tape as well as execute much faster. An American magazine "The Adam Technical Journal" did an excellent issue on this a while ago. Several "fastrun" programs also use this technique. Unfortunately, the Hinkles just touch on it. (Looks like a good topic to explore next issue in Syntax!) But they do give you a good idea of how Basic handles a Basic program.

The Hacker's Guide is 110 pages long. Most of it talks about SmartBasic as we got it from Coleco and that is, as I've tried to make clear, valuable in itself. But the Hinkles went a little better. In exploring Basic, they have found and provided correcting explanations and programs to some of the bugs left in Basic by Coleco. Some of these we know about. the DATA,REM bug and changing Basic on disk so that it looks for its HELLO file properly. They have also offered improvements, such as allowing the LOAD command to "chain" subsequently loaded programs together rather than having the second one write over the first one. (You can change the LOAD command back if you want). They also show you how to make a 40 column screen (Syntax 1.4), define your Smart and command keys, and for the hardware buffs give you some Adam schematics, unfortunately not too clear.

The Hinkles provide a list of documented and undocumented commands as well as some new commands for sprites and sound in a HELLO file (along with most of their other changes and improvements). Note that most of these changes can be written directly onto the disk instead of being stored on a HELLO file. This has the advantage of freeing up space below that which the LOMEM reserves in the HELLO program. The loss will be that of a few unused commands.

The Guide is a readable technical manual for Basic, something that Coleco should have come out with at the very beginning. This is a thorough, the definitive Basic, explanation that provides as much information as it suggests applications.

Program Review: Super Subroc

The original Subroc is an action battle game where enemy ships attacking from the air and under water are to be confronted and destroyed. The player views and controls the action from the perspective of a submarine parascope. Super Subroc offers the same challenge with a much more sophisticated presentation. The super version is more interesting to play because of the superior graphics and sounds and most of all due to the added underwater maneuverability. This gives the game a new dimension as actual underwater battle is possible.

In Subroc you have two firing positions indicated by the two different firing sounds. When attacking the ships in the air, the firing sound is higher in pitch in comparison to the the softer sound when firing torpedoes. Super Subroc offers much more in terms of sound effects. Even the introductory screen has a unique zany tune in addition to the eye-catching falling water drops. During the dive maneuvers, the sound of splashing water is well-simulated. Underwater, the submarine sound is different from that abovewater. If there is a net underwater prohibiting the surface maneuver, it is indicated by the slightly different submarine sound. Finally, when the pause feature is activated, the sound of splashing water and a pitsicatto type music play.

In Subroc, the screen becomes the parascope. There is a vertical position indicator and a horizontal scale that does not seem to serve much purpose. The player quickly loses interest when playing the game because action takes places in one main screen with plenty of repetition. Another drawback is the absence of continuous fire when either side button is held down.

Super Subroc displays a nicer format. The vertical position indicator has a larger scale that is more practical since the submarine can dive. The horizontal scale can be used to judge the horizontal distance but its use is very limited. Short status messages appear at the bottom of the screen. Continuous firing is not possible in this version either. One common characteristic with all super games is the Hall of Fame. Once again, the programmers have done a super job in this area.

Since the original Subroc had to be tailored for a system with more limited memory, this extended version with its dive and surface maneuvers is far superior. Underwater, the task remains very similar to that abovewater except that the alien ships change. Also present are black balls with spikes being held down by a rope. These balls grow bigger and bigger until they finally become gigantic and dangerous. To obtain bonus points, the player must destroy a ship that looks like an octopus. All eight legs must be shot individually. Here is a good graphic demonstration!

In general, super games are much more fun to play than the originals because of their extended screens and added features. Super Subroc is no exception. It offers many refinements which will help the player enjoy and rediscover the game completely.

```
10 &  FAMaccount
12 &  Jeff Benett
14 HOME
16 HTAB 6: INPUT "Date: "; date$: ?
18 ? "For deposit enter letter d     For withdrawl enter w."
20 ?: INPUT "Are you making a deposit or a withdrawl? "; dtwl$:?:?
22 IF dtwl$ <> "d" AND dtwl$ <> "w" THEN 20
24 IF dtwl$ = "d" THEN GOSUB 60
26 IF dtwl$ = "w" THEN GOSUB 80
28 HOME:VTAB 3:INPUT "DO YOU HAVE OTHER TRANSACTIONS (y/n)? ";yes$
30 IF yes$ = "y" THEN 20
32 ?: INPUT "Do you want to activate your   printer (y/n)? "; p$
34 FOR x = 1 TO 800: NEXT: HOME
36 IF p$ = "y" THEN PR #1
38 ? TAB(10); date$:?:?
40 ? TAB(5); "WITHDRAWLS"; TAB(20); "DEPOSITS":?
42 FOR i = i TO w
44 ? TAB(5); "$"; c(i); TAB(21); "$"; m(i):NEXT i
46 ?:?:? TAB(7); "Balance: $"; bal
50 PR #0: END
60 ?:INPUT "What is the amount in dollars  and cents: "; m(w): ?:?
62 bal = bal+m(w)
64 w = w+1
66 INPUT "Another deposit (y/n)? "; yes$:?:?
68 IF yes$ = "y" THEN 60
70 RETURN
80 ?:INPUT "What is the amount in dollars  and cents "; c(w): ?:?
82 bal = bal-c(w)
84 w = w+1
86 INPUT "Another withdrawl (y/n)? "; yes$:?:?
88 IF yes$ = "y" THEN 80
90 RETURN
```

```
10 &  Hex-Rose
11 &
12 TEXT
14 INPUT "HCOLOR: E to W (0-15): "; h
16 INPUT "HCOLOR: N to S (0-15): "; j
18 INPUT "HCOLOR: Border (0-15): "; k
20 HGR2
22 sc = 1.16
25 HCOLOR = j: cx = 140: cy = 96
30 c = COS(.1): s = SIN(.1)
35 &   polar "rose"
40 FOR th = 0 TO 6.3 STEP .05
45 r = 88*COS(2*th)
50 HCOLOR =h:IF r<0 THEN HCOLOR =j
55 x=cx+sc*r*COS(th): y=cy+r*SIN(th)
60 HPLOT 140, 96 TO x, y:NEXT th
65 &   boundary circles
70 HCOLOR = k
75 FOR r=90 TO 95 STEP .8: x=r: y=0
80 FOR i = 0 TO 63
85 t = x*c-y*s: y=x*s+y*c: x=t
90 IF i=0 THEN HPLOT cx+sc*x, cy-y
92 HPLOT TO cx+sc*x, cy-y
94 NEXT i, r
96 GET key$: TEXT
```

```
10 &  Soaring
12 INPUT "HCOLOR = "; k
14 HGR
16 HCOLOR = k
18 f = 1.212: n = 16
20 DIM x(n), y(n)
22 FOR i = 0 TO n
24 READ x(i): NEXT i
26 FOR i = 0 TO n
28 READ y(i)
30 y(i) = -y(i)/f
32 NEXT i
34 HPLOT 140+x(0),
          80+y(0)
36 FOR i = 1 TO n
38 HPLOT TO 140+x(i),
            80+y(i)
40 NEXT i
42 DATA -3,10,30,13,4,
   14,6,0,-14,-12,-20,
   -7,-11,-12,-11,-9,-3
44 DATA 7,24,16,18,0,-6,
   -13,-4,-13,-39,-7,3,
   7,9,10,10,7
46 GET key$: TEXT
```

---------- F C A U G PRODUCT LIST ----------

E-Z EDIT 1.0	$15.00	[ ]
E-Z COPY 1.0	$15.00	[ ]
FCAUG BASIC LIBRARY #1	$10.00	[ ]
FCAUG BASIC LIBRARY #2	$10.00	[ ]
FCAUG BASIC LIBRARY #3	$10.00	[ ]
FCAUG BASIC LIBRARY #4 (avail. April)	$10.00	[ ]
CP/M 2.2 COMMUNICATIONS (MEX, MODEM 7, ADAMBOOT)	$10.00	[ ]
JEOPARDY	$10.00	[ ]
SUPER SUBROC	$10.00	[ ]
TROLL'S TALE	$10.00	[ ]
HARD HAT MACK/PINBALL CONSTRUCTION	$10.00	[ ]
SMARTBASIC 2.0	$10.00	[ ]
ADAMLINK 2	$10.00	[ ]
DATA DRIVE SPEED TESTER	$10.00	[ ]

THIRD PARTY SOFTWARE

ELECTRONIC GAME PACK 1 - by APE Software	$20.00	[ ]
ELECTRONIC GAME PACK 2 - by APE Software	$20.00	[ ]
HACKER'S GUIDE TO ADAM VOL. 1 w/tape or disk	$20.00	[ ]
HACKER'S GUIDE TO ADAM VOL. 2 w/tape or disk	$20.00	[ ]
TIMEX - Time Display Program by C. Hills	$10.00	[ ]
BSR - Block Transmitting Modem Program	$20.00	[ ]
TRS-2-ADAM & TRS-DIR (convert TRS CP/M files)	$20.00	[ ]

ADAM GAME CARTRIDGES

LADYBUG, VENTURE, PEPPER II (Limited Quantity) each	$25.00	[ ]
B.C'S QUEST FOR TIRES	$20.00	[ ]
JUMPMAN JUNIOR	$15.00	[ ]
OMEGA RACE	$15.00	[ ]
CABBAGE PATCH KIDS Adventures in The Park	$15.00	[ ]
FRONT LINE (requires Super Action Controller)	$15.00	[ ]
DESTRUCTOR (requires Driving Module)	$15.00	[ ]

ADAM SOFTWARE (original manuals & media)

SMARTLOGO	$30.00	[ ]
SMARTFILER (Rev. 27D)	$30.00	[ ]
EXPERTYPE	$35.00	[ ]
ADAMCALC	$55.00	[ ]
32 BASIC PROGRAMS (Book & tape)	$30.00	[ ]
32 BASIC PROGRAMS (Book only)	$20.00	[ ]
CP/M 2.2 & ASSEMBLER (on disk)	$60.00	[ ]

ADAM HARDWARE AND ACCESSORIES

64K MEMORY EXPANDER	$75.00	[ ]
CENTRONICS (PARALLEL) PRINTER INTERFACE	$85.00	[ ]
SMARTBASIC 1.0 on ROM CARTRIDGE	$70.00	[ ]
ADAMLINK MODEM w/ADAMLINK 2 program	$125.00	[ ]
SMARTBASIC 1.0 REPLACEMENT DATA PACKS	$12.00	[ ]
FLIPPY TAPES (both sides are used)	$10.00	[ ]
PICA 10 DAISY WHEEL	$12.00	[ ]
For Hardware items, please add $3 for postage and handling.		[ ]

FCAUG Box 547, Victoria Station, Westmount, Que. H3Z 2Y6
