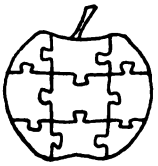

Apple Assembly Line

Volume 1 -- Issue 7 April, 1981

As of today the total distribution of the Apple Assembly Line is nearly 350. Let's shoot for 1000 by the end of 1981! I will have a full page ad in the next eight issues of NIBBLE, so I think 1000 is a reasonable goal. Thank you for your support!

In This Issue...

Cross Reference (XREF) for S-C ASSEMBLER II

Bob Kovacs has a new product, one which many of you have asked me for. It enables you to produce a complete cross reference listing of all symbols used in an assembly language program. See his ad on page 7 for a description and ordering information.

I am honored to have three companies (Rak-Ware, Decision Systems, and Flatland Software) producing software to complement my assembler!

80 Columns on Your Printer

For some reason unknown to me Apple's Parallel Interface Card comes with at least three different ROM's. There seems to me no indication on the package which one you are getting, and no listing in the manual of the exact ROM on the board. This leads to confusion, because some ROM versions will print 80-column assembly listings at the drop of a hat (Just type PR#1 and ASM, and you have it!); but others require special treatment.

If you have the latter type, I have found that this works:

```
:PR#1          (assuming slot # 1)
:$579:50       ($578 + slot#     )
:ASM
```

Text File I/O in Assembly Language Programs

A surprisingly large number of people have written or called to ask the same question:

> "How can I read or write a text file from my program? I know I can issue OPEN, READ, WRITE, and CLOSE commands just like in Applesoft -- by outputting a control-D and the command string. But after that, where is the data?"

It is really very simple, and after I tell you, you may be just as embarrassed as they were!

Remember that in Applesoft, after opening a file and setting it up to read with the OPEN and READ commands, you actually read it with normal INPUT statements. In assembly language you do the same thing. You can either input a line by calling the monitor routine at $FD6F, or you can read character-by-character by calling the character input routine at $FD0C. After a JSR $FD0C, the input character will be in the A-register. After a JSR $FD6F, the input line will be in the monitors buffer starting at $0200, and the X-register will contain the number of characters in the line (not counting the carriage return).

Also remember that after using the OPEN and WRITE commands, all you do in Applesoft to write on a text file is use the normal PRINT statement. In the same way, from assembly language, you just call the monitor print character routine at $FDED. The character to be written should be in the A-register, and then use JSR $FDED.

Here is a little program which opens a text file and reads it into a buffer at $4000. It demonstrates a few more tricks you might need to know, as well.

Lines 1180-1270 patch DOS so that it thinks you are executing an Applesoft program. (If you really are calling this from a RUNning Applesoft program, you can skip lines 1190 and 1200.) We want to be able to issue DOS commands by printing control-D and the command string, so we have to be RUNning. We want to be able to tell when the end-of-file comes without getting an "OUT OF DATA" error, so we turn on the Applesoft ON ERR flag and set it up to branch to our own END.OF.DATA routine.

Lines 1310-1350 print the DOS OPEN and READ commands. The message printer is a very simple loop at lines 1630-1690.

Lines 1380-1500 read the characters from the file and store them in a buffer at $4000. I save the stack pointer before the loop so I can restore it after the end-of-file occurs. Lines 1530-1570 restore the stack pointer, close the file, and return to DOS.

I really should clean up the mess I created with lines 1180-1270, but I will leave that as an exercise for the reader.

```
                        1000 *-------------------------------
                        1010 *       DEMONSTRATION OF READING A TEXT FILE
                        1020 *-------------------------------
0033-                   1030 PROMPT.CHAR          .EQ $33
0075-                   1040 CURRENT.LINE.NO      .EQ $75,76
009D-                   1050 BUF.PNTR             .EQ $9D,9E
AAB6-                   1060 DOS.LANGUAGE.FLAG    .EQ $AAB6
00D8-                   1070 ONERR.FLAG           .EQ $D8
9D5A-                   1080 DOS.ONERR.PNTR       .EQ $9D5A,9D5B
03D0-                   1090 DOS.REENTRY          .EQ $3D0
FD0C-                   1100 MON.RDKEY            .EQ $FD0C
FDED-                   1110 MON.COUT             .EQ $FDED
                        1120 *-------------------------------
                        1130 TEXT.READER
                        1140 *-------------------------------
                        1150 *       PATCH DOS SO END OF FILE WILL
                        1160 *       BRANCH TO MY "END.OF.DATA"
                        1170 *-------------------------------
0800- A9 01             1180         LDA #1       TELL DOS WE ARE IN APPLESOFT
0802- 8D B6 AA          1190         STA DOS.LANGUAGE.FLAG
0805- 85 76             1200         STA CURRENT.LINE.NO+1  NOT IN DIRECT MODE
0807- 85 33             1210         STA PROMPT.CHAR        NOT DIRECT MODE
0809- A9 FF             1220         LDA #$FF     TURN ON "ON ERR"
080B- 85 D8             1230         STA ONERR.FLAG
080D- A9 3C             1240         LDA #END.OF.DATA
080F- 8D 5A 9D          1250         STA DOS.ONERR.PNTR
0812- A9 08             1260         LDA /END.OF.DATA
0814- 8D 5B 9D          1270         STA DOS.ONERR.PNTR+1
                        1280 *-------------------------------
                        1290 *       OPEN THE FILE
                        1300 *-------------------------------
0817- A0 00             1310         LDY #QOPEN-QTS
0819- 20 48 08          1320         JSR QUOTE.PRINT
081C- A0 10             1330         LDY #QREAD-QTS
081E- 20 48 08          1340         JSR QUOTE.PRINT
                        1350 *-------------------------------
                        1360 *       READ THE FILE
                        1370 *-------------------------------
0821- BA                1380         TSX
0822- 8E 7C 08          1390         STX OLD.STACK.PNTR
0825- A9 00             1400         LDA #BUFFER
0827- 85 9D             1410         STA BUF.PNTR
0829- A9 40             1420         LDA /BUFFER
082B- 85 9E             1430         STA BUF.PNTR+1
082D- 20 0C FD          1440 .1      JSR MON.RDKEY   READ CHARACTER
0830- A0 00             1450         LDY #0
0832- 91 9D             1460         STA (BUF.PNTR),Y
0834- E6 9D             1470         INC BUF.PNTR
0836- D0 F5             1480         BNE .1
0838- E6 9E             1490         INC BUF.PNTR+1
083A- D0 F1             1500         BNE .1       ...ALWAYS
                        1510 *-------------------------------
                        1520 END.OF.DATA
083C- AE 7C 08          1530         LDX OLD.STACK.PNTR
083F- 9A                1540         TXS
0840- A0 20             1550         LDY #QCLOSE-QTS
0842- 20 48 08          1560         JSR QUOTE.PRINT
0845- 4C D0 03          1570         JMP DOS.REENTRY
                        1580 *-------------------------------
                        1590 *       PRINT A MESSAGE
                        1600 *       MESSAGE STARTS AT QTS,Y
                        1610 *       MESSAGE ENDS WITH 00 BYTE
                        1620 *-------------------------------
                        1630 QUOTE.PRINT
0848- B9 54 08          1640 .1      LDA QTS,Y
084B- F0 06             1650         BEQ .2
084D- 20 ED FD          1660         JSR MON.COUT
0850- C8                1670         INY
0851- D0 F5             1680         BNE .1       ...ALWAYS
0853- 60                1690 .2      RTS
                        1700 *-------------------------------
0854-                   1710 QTS     .EQ *
0854- 84                1720 QOPEN   .HS 84       CONTROL-D
0855- CF D0 C5
0858- CE A0 D4
085B- C5 D3 D4
085E- C6 C9 CC
0861- C5                1730         .AS -/OPEN TESTFILE/
0862- 8D 00             1740         .HS 8D00
```

```
0864- 84        1750 QREAD   .HS 84        CONTROL-D
0865- D2 C5 C1
0868- C4 A0 D4
086B- C5 D3 D4
086E- C6 C9 CC
0871- C5        1760         .AS -/READ TESTFILE/
0872- 8D 00     1770         .HS 8D00
0874- 84        1780 QCLOSE  .HS 84        CONTROL-D
0875- C3 CC CF
0878- D3 C5     1790         .AS -/CLOSE/
087A- 8D 00     1800         .HS 8D00
                1810 *-------------------------------
087C-           1820 OLD.STACK.PNTR .BS 1
                1830 *-------------------------------
4000-           1840 BUFFER      .EQ $4000
                1850 *-------------------------------
```

SYMBOL TABLE

```
009D- BUF.PNTR
4000- BUFFER
0075- CURRENT.LINE.NO
AAB6- DOS.LANGUAGE.FLAG
9D5A- DOS.ONERR.PNTR
03D0- DOS.REENTRY
083C- END.OF.DATA
FDED- MON.COUT
FD0C- MON.RDKEY
087C- OLD.STACK.PNTR
00D8- ONERR.FLAG
0033- PROMPT.CHAR
0874- QCLOSE
0854- QOPEN
0864- QREAD
0854- QTS
0848- QUOTE.PRINT
.01=0848, .02=0853
0800- TEXT.READER
.01=082D
```

:

Applesoft Internal Entry Points

An excellent article appeared just over a year ago (by the same title) in The Apple Orchard, Volume 1, Number 1, March/April 1980. John Crossley of Apple Computer, Inc. wrote it. He revealed most of the usable entry points within the Applesoft ROM, and many details on how they work and how to use them. If you don't have that magazine, go get one right away. They are available at some stores, through some local Apple clubs, and directly from the publisher (the Internatioal Apple Corps). There are a few typographical errors, but you should be able to figure them out by comparing with a disassembly.

To get you started, I have made up a list of my own which includes the starting addresses for all the keyword routines. I got these from the ROM itself. The keyword list starts at $D0D0, and a parallel list of addresses starts at $D000. The addresses in the list are all low-byte-first, and are all pointing to one byte before the actual start. That is because Applesoft branches to the appropriate routine by placing the address from this list on the stack and then using RTS (see AAL issue #1, page 11, for an explanation of this technique).

Applesoft Entry Points

This chart shows all the token values for Applesoft, and the address where the token is processed.

token		keyword	addr
80	128	END	D870
81	129	FOR	D766
82	130	NEXT	DCF9
83	131	DATA	D995
84	132	INPUT	DBB2
85	133	DEL	F331
86	134	DIM	DFD9
87	135	READ	DBE2
88	136	GR	F390
89	137	TEXT	F399
8A	138	PR#	F1E5
8B	139	IN#	F1DE
8C	140	CALL	F1D5
8D	141	PLOT	F225
8E	142	HLIN	F232
8F	143	VLIN	F241
90	144	HGR2	F3D8
91	145	HGR	F3E2
92	146	HCOLOR=	F6E9
93	147	HPLOT	F6FD
94	148	DRAW	F769
95	149	XDRAW	F76F
96	150	HTAB	F7E7
97	151	HOME	FC58
98	152	ROT=	F721
99	153	SCALE=	F727
9A	154	SHLOAD	F775
9B	155	TRACE	F26D
9C	156	NOTRACE	F26F
9D	157	NORMAL	F273
9E	158	INVERSE	F277
9F	159	FLASH	F280
A0	160	COLOR=	F24F
A1	161	POP	D96B
A2	162	VTAB	F256
A3	163	HIMEM:	F286
A4	164	LOMEM:	F2A6
A5	165	ONERR	F2CB
A6	166	RESUME	F318
A7	167	RECALL	F3BC
A8	168	STORE	F39F
A9	169	SPEED=	F262
AA	170	LET	DA46
AB	171	GOTO	D93E
AC	172	RUN	D912
AD	173	IF	D9C9
AE	174	RESTORE	D849
AF	175	&	03F5
B0	176	GOSUB	D921
B1	177	RETURN	D96B
B2	178	REM	D9DC
B3	179	STOP	D86E
B4	180	ON	D9EC
B5	181	WAIT	E784
B6	182	LOAD	D8C9
B7	183	SAVE	D8B0
B8	184	DEF	E313
B9	185	POKE	E77B
BA	186	PRINT	DAD5
BB	187	CONT	D896
BC	188	LIST	D6A5
BD	189	CLEAR	D66A
BE	190	GET	DBA0
BF	191	NEW	D649
C0	192	TAB(	
C1	193	TO	
C2	194	FN	
C3	195	SPC(	
C4	196	THEN	
C5	197	AT	
C6	198	NOT	
C7	199	STEP	
C8	200	+	
C9	201	-	
CA	202	*	
CB	203	/	
CC	204	^	
CD	205	AND	
CE	206	OR	
CF	207	>	
D0	208	=	
D1	209	<	
D2	210	SGN	EB91
D3	211	INT	EC24
D4	212	ABS	EBB0
D5	213	USR	000A
D6	214	FRE	E2DF
D7	215	SCRN(	D413
D8	216	PDL	DFCE
D9	217	POS	E300
DA	218	SQR	EE8E
DB	219	RND	EFAF
DC	220	LOG	E942
DD	221	EXP	EF0A
DE	222	COS	EFEB
DF	223	SIN	EFF2
E0	224	TAN	F03B
E1	225	ATN	F09F
E2	226	PEEK	E765
E3	227	LEN	E6D7
E4	228	STR$	E3C6
E5	229	VAL	E708
E6	230	ASC	E6E6
E7	231	CHR$	E647
E8	232	LEFT$	E65B
E9	233	RIGHT$	E687
EA	234	MID$	E691

Patch S-C Assembler II for More Errors

Some of you have asked for a way to see all your errors at once. If you patch Version 4.0 in this simple way, you will see all error messages during one ASM, instead of aborting the assembly after the first error.

Look at $1752 to $1754; you should see 20 81 1A. If you do, then make this patch:

```
:$1752:4C 8E 18
```

Now try an assembly of some source code with several errors in it. You will see all the errors on your screen. Or if your printer is on, they will all print.

Personally, I liked it better the other way. But if you never make more than one error per program, you won't be able to tell the difference!

Fast String Input Routine for Applesoft

Yet another use for the imperious ampersand! This program will read a line from the keyboard or a text file into a string variable. It will accept commas and colons without complaint, too. No more "EXTRA IGNORED" messages, and much less chance of garbage collection tying things up.

The program is shown here with the origin set to $0300, the most popular place in your Apple. If that taxi is already full, you can change the origin to whatever you like. In fact, the subroutine itself is completely relocatable. You can put it anywhere in memory you like, just so you set $3F6 and 3F7 to point to it.

Lines 1160-1220 are executed if you BRUN a file with this program on it. They put a JMP GET into $3F5, so that the "&" will call my subroutine. Once this code is executed, you can execute statements like "&GET A$" to read a line into a string.

Lines 1240-1500 are the input subroutine. At line 1240 the token following the ampersand is tested; it should be $BE, which is the token for "GET". If not, JMP $DEC9 makes your screen say "SYNTAX ERROR"!

Lines 1270 and 1280 set up the address of the string variable in locations $83 and $84. We will use this later to tell Applesoft where the input line is.

Lines 1290-1360 change the prompt symbol to a bell (in case you backspace too much) and call on the monitor input routine to read a line. After the line is read, the prompt is restored to whatever it was before. The length of the input line is in the X-register, and the line itself is in the buffer starting at $0200.

Lines 1370 and 1380 call on Applesoft to set aside space for the input line in the string area. This may force garbage collection if you are about out of memory at the time. GETSPA leaves the address of the start of the slot set aside for our input line in locations $71 and $72.

Lines 1390-1460 store the length and address of the input line into the string variable. The address is of the slot GETSPA just reserved.

Lines 1470-1500 call on MOVSTR to copy the input line from the monitor's input buffer (at $0200) into the slot reserved by GETSPA.

Now if you want to read some data off the disk which might have commas and colons in it, you can do it like this:

```
100 PRINT CHR$(4) "OPEN MY.FILE"
110 PRINT CHR$(4) "READ MY.FILE"
120 FOR I = 1 TO 10
130 & GET A$(I)
140 NEXT I
```

```
                    1000 *---------------------------------
                    1010 *      FAST INPUT STRING ROUTINE
                    1020 *      &GET <STRING VARIABLE>
                    1030 *      ACCEPTS ANY CHARACTER, UNLIKE NORMAL INPUT
                    1040 *---------------------------------
03F5-               1050 AMPERSAND.VECTOR    .EQ $3F5
009D-               1060 LENGTH .EQ $9D
DEC9-               1070 SYNTAX.ERROR .EQ $DEC9
DFE3-               1080 PTRGET .EQ $DFE3
E452-               1090 GETSPA .EQ $E452
E5E2-               1100 MOVSTR .EQ $E5E2
                    1110 *---------------------------------
0033-               1120 MON.PROMPT .EQ $33
FD6F-               1130 MON.RDLINE .EQ $FD6F
                    1140 *---------------------------------
                    1150         .OR $300
0300- A9 4C         1160         LDA #$4C     JUMP INSTRUCTION
0302- 8D F5 03      1170         STA AMPERSAND.VECTOR
0305- A9 10         1180         LDA #GET
0307- 8D F6 03      1190         STA AMPERSAND.VECTOR+1
030A- A9 03         1200         LDA /GET
030C- 8D F7 03      1210         STA AMPERSAND.VECTOR+2
030F- 60            1220         RTS
                    1230 *---------------------------------
0310- C9 BE         1240 GET     CMP #$BE     GET TOKEN
0312- F0 03         1250         BEQ .1       YES
0314- 4C C9 DE      1260         JMP SYNTAX.ERROR
0317- 20 B1 00      1270 .1      JSR $B1
031A- 20 E3 DF      1280         JSR PTRGET   GET STRING DESCRIPTOR
031D- A5 33         1290         LDA MON.PROMPT
031F- 48            1300         PHA
0320- A9 87         1310         LDA #$87     BELL FOR PROMPT
0322- 85 33         1320         STA MON.PROMPT
0324- 20 6F FD      1330         JSR MON.RDLINE   INPUT A LINE
0327- 68            1340         PLA
0328- 85 33         1350         STA MON.PROMPT
032A- 86 9D         1360         STX LENGTH   SAVE LENGTH
032C- 8A            1370         TXA
032D- 20 52 E4      1380         JSR GETSPA   GET SPACE IN STRING AREA
0330- A0 00         1390         LDY #0       MOVE DATA INTO VARIABLE
0332- 91 83         1400         STA ($83),Y  LENGTH
0334- A5 71         1410         LDA $71
0336- C8            1420         INY
0337- 91 83         1430         STA ($83),Y  LO-BYTE OF ADDRESS
0339- A5 72         1440         LDA $72
033B- C8            1450         INY
033C- 91 83         1460         STA ($83),Y  HI-BYTE OF ADDRESS
033E- A0 02         1470         LDY /$200    SET UP TO COPY STRING DATA
0340- A2 00         1480         LDX #$200    INTO STRING AREA
0342- A5 9D         1490         LDA LENGTH
0344- 4C E2 E5      1500         JMP MOVSTR   COPY IT NOW
```

SYMBOL TABLE

```
03F5- AMPERSAND.VECTOR          0033- MON.PROMPT
0310- GET                       FD6F- MON.RDLINE
.01=0317                        E5E2- MOVSTR
E452- GETSPA                    DFE3- PTRGET
009D- LENGTH                    DEC9- SYNTAX.ERROR
```

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203
817/382-6353

DIS-ASSEMBLER

DSA-DS dis-assembles Apple machine language programs into forms compatible with LISA, S-C ASSEMBLER (3.2 or 4.0), Apple's TOOL-KIT ASSEMBLER and others. DSA-DS dis-assembles instructions or data. Labels are generated for referenced locations within the machine language program.
$25, Disk, Applesoft (32K, ROM or Language card)

OTHER PRODUCTS

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your **BASIC** programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Use advanced logic constructs such as **IF...ELSE...**, **CASE**, **SELECT**, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of **PASCAL**.
$35. Disk, Applesoft (48K, ROM or Language Card).

FORM-DS is a complete system for the definition of input and output froms. **FORM-DS** supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
$25 Disk, Applesoft (32K, ROM or Language Card).

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's **CLEAR** gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
$25 Disk, Applesoft.

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, **SPEED-DS** includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
$15 Disk, Applesoft (32K, ROM or Language Card).

(Add $4.00 for Foreign Mail)

Hiding Things Under DOS.....Rick Hatcher

In issue number 5/1980 of NIBBLE, a small article by William Reynolds III tells how to do something I have wondered about for a long time. That is how to move the HIMEM pointer down so that machine language code or something else can be put out of the way and protected. For example: I have a lower-case routine I like to use on key input; I also like to use the character display routine from Lawrence Hall of Science which is hooked into the control-Y pointer. This is one way to dump memory in both hex and ASCII. I have looked for protected areas but until now the only place seemed to be from $300 to $3CF. This is a little over 200 bytes, and I needed about 400.

Neil Konzen's Program Line Editor (from Call A.P.P.L.E.) moves the file buffers down and leaves space between the buffers and DOS...but the manual which I sneaked a look at does not tell how to do it. The article in NIBBLE on page 40 finally revealed the secret. The file buffers are located by a pointer at locations $9D00 and $9D01 (least significant byte first, as usual). A DOS routine at $A7D4 builds the buffers using this pointer and the value of MAXFILES (at $AA57). [note: all addresses assume a 48K system]

All you have to do is change the address at $9D00.9D01 and call the routine at $A7D4. I wanted to create a space of $200 bytes (512 decimal). The normal value at $9D00.9D01 is $9CD3. I changed it to $9AD3, and then typed A7D4G in the monitor. The value of HIMEM was automatically changed to $9400 from the usual $9600. The protected area is from $9B00 to $9CFF. The buffers are located from $9400 to $9AFF and DOS is located from $9D00 to BFFF. If a MAXFILES command is used it changes HIMEM but the buffer top at $9AFF stays unchanged.

To make space like this from an Applesoft program. here is all you need:

```
100 POKE 40193,154
110 POKE 40194,211
120 CALL 42964
```

It isn't so easy in Integer BASIC, because the routine moves HIMEM without moving the program down in memory. (Remember Integer BASIC programs are at the top of memory up against HIMEM; Applesoft programs are at the low end of memory.) The NIBBLE article gives a method for Integer BASIC, but I haven't tried it.

I use an Applesoft HELLO program which first does the three lines above, and then BRUNs or BLOADs the code I want to hide. The BRUN portion sets up the I/O hooks at $36.39 and sets up the control-Y vector at $3F8. I use the BLOAD if I want the code resident but not hooked in.

Once the space is made, it stays there. If you INIT a slave disk, the slave has the same change.

The NIBBLE article reveals a few more details about the buffers in which you may be interested.

Commented Listing of DOS 3.2.1 Format

Here is the second installment of DOS disassembly, covering the area from $BEA0 through $BFFF. If you read the listing in last month's AAL carefully. you probably noted that it ended with the label definition "FORMAT", but no code followed. Well, here it is!

FORMAT turns a blank diskette into one with address headers recorded on every track. Otherwise, the disk is empty. No directory is written into track $11 yet, nor is any DOS recorded yet in tracks 0, 1, and 2. When you use the INIT command, the first step exectured is to format the disk; after formatting, a DOS image and empty directory are written; then your HELLO program is SAVEd.

The Apple Disk Interface depends on critical software timing to operate correctly. You will find many strange sequences of code (such as PHA, PLA, NOP, PHA, PLA between $BF47 and $BF4B) which are for timing purposes. If you are interested in counting cycles, the timing for each opcode-address mode combination are listed in the Quick Reference Card that came with your S-C ASSEMBLER II Version 4.0.

```
                    1000 *       .LIST OFF
                    1010 *-------------------------------
                    1020 *       DOS 3.2.1 DISASSEMBLY $BEA0-BFFF
                    1030 *       BOB SANDER-CEDERLOF      3-26-81
                    1040 *-------------------------------
0478-               1050 CURRENT.TRACK       .EQ $478
                    1060 *-------------------------------
C080-               1070 PHASE.OFF           .EQ $C080
C081-               1080 PHASE.ON            .EQ $C081
C088-               1090 MOTOR.OFF           .EQ $C088
C089-               1100 MOTOR.ON            .EQ $C089
C08A-               1110 ENABLE.DRIVE.1      .EQ $C08A
C08B-               1120 ENABLE.DRIVE.2      .EQ $C08B
C08C-               1130 Q6L                 .EQ $C08C
C08D-               1140 Q6H                 .EQ $C08D
C08E-               1150 Q7L                 .EQ $C08E
C08F-               1160 Q7H                 .EQ $C08F
                    1170 *-------------------------------
002D-               1180 SECTOR      .EQ $2D
002F-               1190 VOLUME      .EQ $2F
0041-               1200 TRACK.CNTR  .EQ $41
0046-               1210 DATA.CNTR   .EQ $46
0047-               1220 SYNC.CNT    .EQ $47
004A-               1230 CONST.AA    .EQ $4A
004B-               1240 FILL.CNTR   .EQ $4B
004B-               1250 FMT.SECTOR  .EQ $4B
                    1260 *-------------------------------
B965-               1270 READ.ADDRESS        .EQ $B965
BA1E-               1280 SEEK.TRACK.ABSOLUTE .EQ $BA1E
BE37-               1290 RWTS.EXIT           .EQ $BE37
BE39-               1300 ERROR.HANDLER       .EQ $BE39
                    1310 *-------------------------------
0040-               1320 ERR.BAD.DRIVE       .EQ $40
                    1330 *-------------------------------
                    1340         .OR $BEA0
                    1350         .TA $800
                    1360 *-------------------------------
BEA0- A9 80         1370 FORMAT  LDA #128     SET CURRENT TRACK REAL HIGH
BEA2- 8D 78 04      1380         STA CURRENT.TRACK   SO DRIVE WILL HOME
BEA5- A9 00         1390         LDA #0              TO TRACK 0
BEA7- 85 41         1400         STA TRACK.CNTR  INIT COUNTER FOR INIT ROUTINE
BEA9- 20 1E BA      1410         JSR SEEK.TRACK.ABSOLUTE
                    1420 *-------------------------------
BEAC- A9 AA         1430         LDA #$AA     SAVE $AA IN PAGE ZERO FOR TIMING
BEAE- 85 4A         1440         STA CONST.AA
```

```
                        1450 *-------------------------------
                        1460 *      FILL ENTIRE TRACK WITH SYNC BYTES
                        1470 *-------------------------------
BEB0- A0 50             1480         LDY #80       START WITH 80 SYNC-BYTES
                        1490 FILL.TRACK.WITH.SYNC
BEB2- 84 47             1500         STY SYNC.CNT  # OF SYNC BYTES BETWEEN SECTORS
BEB4- A9 27             1510         LDA #39       WRITE SYNC'S OVER ENTIRE TRACK
BEB6- 85 4B             1520         STA FILL.CNTR
BEB8- BD 8D C0          1530         LDA Q6H,X     GET READY TO WRITE
BEBB- BD 8E C0          1540         LDA Q7L,X
BEBE- A9 FF             1550         LDA #$FF      WRITE $FF EVERYWHERE
BEC0- 9D 8F C0          1560         STA Q7H,X     ALL SET TO WRITE....
BEC3- DD 8C C0          1570         CMP Q6L,X
BEC6- 24 00             1580         BIT $00       DELAY 3 CYCLES
BEC8- 88                1590 .1      DEY
BEC9- F0 0F             1600         BEQ .3
BECB- 48                1610         PHA
BECC- 68                1620         PLA           THESE ARE JUST FOR TIMING
BECD- EA                1630         NOP           NEED 27 CYCLES BTWN WRITES
BECE- 48                1640 .2      PHA
BECF- 68                1650         PLA
BED0- EA                1660         NOP
BED1- EA                1670         NOP
BED2- 9D 8D C0          1680         STA Q6H,X     WRITE SYNC BYTE
BED5- DD 8C C0          1690         CMP Q6L,X
BED8- B0 EE             1700         BCS .1        ...ALWAYS
BEDA- C6 4B             1710 .3      DEC FILL.CNTR  TRACK FULL YET?
BEDC- D0 F0             1720         BNE .2        NO
                        1730 *-------------------------------
                        1740 *      WRITE 13-SECTOR HEADERS ON TRACK
                        1750 *
                        1760 *          EACH SECTOR CONSISTS OF AN ADDRESS BLOCK
                        1770 *          AND A DATA BLOCK.
                        1780 *          ADDRESS:  D5 AA B5 V1 V2 T1 T2
                        1790 *                    S1 S2 C1 C2 DE AA EB
                        1800 *          DATA:     FORMATTED TO ALL SYNC BYTES
                        1810 *-------------------------------
                        1820 FORMAT.TRACK
BEDE- A4 47             1830         LDY SYNC.CNT  # SYNC BYTES BTWN SECTORS
BEE0- EA                1840         NOP
BEE1- EA                1850         NOP
BEE2- D0 06             1860 .1      BNE .4        ...ALWAYS
                        1870 *-------------------------------
BEE4- 48                1880 .2      PHA           WRITE SYNC BYTES BEFORE SECTOR
BEE5- 68                1890         PLA
BEE6- 48                1900         PHA
BEE7- 68                1910         PLA
BEE8- C1 00             1920         CMP ($00,X)   DELAY 6 CYCLES
BEEA- EA                1930 .4      NOP
BEEB- 9D 8D C0          1940 .5      STA Q6H,X     WRITE NEXT SYNC BYTE
BEEE- DD 8C C0          1950         CMP Q6L,X
BEF1- 88                1960         DEY
BEF2- D0 F0             1970         BNE .2
                        1980 *-------------------------------
BEF4- A9 D5             1990         LDA #$D5      WRITE D5 AA B5
BEF6- 20 CC BF          2000         JSR WRITE.BYTE.2
BEF9- A9 AA             2010         LDA #$AA
BEFB- 20 CD BF          2020         JSR WRITE.BYTE.3
BEFE- A9 B5             2030         LDA #$B5
BF00- 20 CD BF          2040         JSR WRITE.BYTE.3
BF03- A5 2F             2050         LDA VOLUME    WRITE VOLUME, TRACK, AND SECTOR
BF05- 20 BD BF          2060         JSR WRITE.BYTE.1
BF08- A5 41             2070         LDA TRACK.CNTR
BF0A- 20 BD BF          2080         JSR WRITE.BYTE.1
BF0D- A5 4B             2090         LDA FMT.SECTOR
BF0F- 20 BD BF          2100         JSR WRITE.BYTE.1
BF12- A5 2F             2110         LDA VOLUME    COMPUTE CHECKSUM
BF14- 45 41             2120         EOR TRACK.CNTR
BF16- 45 4B             2130         EOR FMT.SECTOR
BF18- 48                2140         PHA           WRITE CHECKSUM
BF19- 4A                2150         LSR
BF1A- 05 4A             2160         ORA CONST.AA   #$AA, FOR TIMING
BF1C- 9D 8D C0          2170         STA Q6H,X
BF1F- DD 8C C0          2180         CMP Q6L,X
BF22- 68                2190         PLA
BF23- 09 AA             2200         ORA #$AA
BF25- 20 CC BF          2210         JSR WRITE.BYTE.2
BF28- A9 DE             2220         LDA #$DE      WRITE DE AA EB
BF2A- 20 CD BF          2230         JSR WRITE.BYTE.3
BF2D- A9 AA             2240         LDA #$AA
BF2F- 20 CD BF          2250         JSR WRITE.BYTE.3
```

```
BF32- A9 EB     2260          LDA #$EB
BF34- 20 CD BF  2270          JSR WRITE.BYTE.3
BF37- A9 FF     2280          LDA #$FF        WRITE MORE SYNC BYTES
BF39- 20 CD BF  2290          JSR WRITE.BYTE.3
BF3C- A0 02     2300          LDY #2          FILL WHOLE DATA BLOCK WITH $FF
BF3E- 84 46     2310          STY DATA.CNTR
BF40- A0 AD     2320          LDY #173
BF42- D0 06     2330          BNE .7          ...ALWAYS
BF44- 88        2340 .6       DEY             FINISHED?
BF45- F0 0D     2350          BEQ .8          YES, AT LEAST THIS GROUP
BF47- 48        2360          PHA             23 CYCLES PER BYTE
BF48- 68        2370          PLA
BF49- EA        2380          NOP
BF4A- 48        2390 .7       PHA
BF4B- 68        2400          PLA
BF4C- 9D 8D C0  2410          STA Q6H,X
BF4F- DD 8C C0  2420          CMP Q6L,X
BF52- B0 F0     2430          BCS .6          ...ALWAYS
BF54- C6 46     2440 .8       DEC DATA.CNTR   FINISHED?
BF56- D0 F2     2450          BNE .7          NOT YET, DO SECOND GROUP
                2460 *--------------------------------
BF58- A4 47     2470          LDY SYNC.CNT
BF5A- 18        2480          CLC
BF5B- 24 00     2490          BIT $00         DELAY
BF5D- 9D 8D C0  2500          STA Q6H,X
BF60- BD 8C C0  2510          LDA Q6L,X
BF63- A5 4B     2520          LDA FMT.SECTOR  COMPUTE NEXT SECTOR #
BF65- 69 0A     2530          ADC #10         SKEW FACTOR = 10
BF67- 85 4B     2540          STA FMT.SECTOR
BF69- E9 0C     2550          SBC #12
BF6B- F0 0A     2560          BEQ CHECK.TRACK
BF6D- B0 01     2570          BCS .9          STORE VALUE MODULO 13
BF6F- 2C        2580          .HS 2C          'BIT' OPCODE TO SKIP NEXT TWO BYTES
BF70- 85 4B     2590 .9       STA FMT.SECTOR
BF72- A9 FF     2600          LDA #$FF
BF74- 4C EB BE  2610          JMP .5          DO NEXT SECTOR
                2620 *--------------------------------
                2630 *        CHECK WHETHER TRACK OVERLAPPED
                2640 *--------------------------------
                2650 CHECK.TRACK
BF77- 48        2660          PHA             TIME DELAY
BF78- 68        2670          PLA
BF79- A4 47     2680          LDY SYNC.CNT
BF7B- BD 8D C0  2690          LDA Q6H,X       SET UP TO READ
BF7E- BD 8E C0  2700          LDA Q7L,X       SENSE WRITE PROTECT
BF81- 30 32     2710          BMI .4          DRIVE ERROR
BF83- 88        2720          DEY
BF84- 48        2730 .1       PHA             DELAY LOOP
BF85- 68        2740          PLA
BF86- 48        2750          PHA
BF87- 68        2760          PLA
BF88- 48        2770          PHA
BF89- 68        2780          PLA
BF8A- 88        2790          DEY             FINISHED WITH DELAY YET?
BF8B- D0 F7     2800          BNE .1          NO
BF8D- 20 65 B9  2810          JSR READ.ADDRESS
BF90- B0 04     2820          BCS .2          BAD READ
BF92- A5 2D     2830          LDA SECTOR      SHOULD BE SECTOR 0
BF94- F0 0A     2840          BEQ .3          YES!
BF96- A4 47     2850 .2       LDY SYNC.CNT    DIMINISH SYNC COUNT
BF98- 88        2860          DEY             AND TRY AGAIN
BF99- C0 10     2870          CPY #16         UNLESS NOT ENOUGH LEFT
BF9B- 90 18     2880          BCC .4          DRIVE ERROR
BF9D- 4C B2 BE  2890          JMP FILL.TRACK.WITH.SYNC
                2900 *--------------------------------
BFA0- E6 41     2910 .3       INC TRACK.CNTR  NEXT TRACK
BFA2- A5 41     2920          LDA TRACK.CNTR
BFA4- C9 23     2930          CMP #35         FINISHED?
BFA6- B0 12     2940          BCS .5          YES
BFA8- 0A        2950          ASL             DOUBLE FOR TRACK SEEK ROUTINE
BFA9- 20 1E BA  2960          JSR SEEK.TRACK.ABSOLUTE
BFAC- A4 47     2970          LDY SYNC.CNT    BUMP SYNC.CNT BEFORE TRYING
BFAE- C8        2980          INY             NEXT TRACK
BFAF- C8        2990          INY
BFB0- 84 47     3000          STY SYNC.CNT
BFB2- 4C B2 BE  3010          JMP FILL.TRACK.WITH.SYNC
                3020 *--------------------------------
BFB5- A9 40     3030 .4       LDA #ERR.BAD.DRIVE
BFB7- 4C 39 BE  3040          JMP ERROR.HANDLER
                3050 *--------------------------------
BFBA- 4C 37 BE  3060 .5       JMP RWTS.EXIT
```

```
                        3070 *-------------------------------
                        3080 *       SUBROUTINES TO WRITE BYTE ON DISK
                        3090 *-------------------------------
                        3100 WRITE.BYTE.1
BFBD- 48                3110         PHA            ADDRESS BLOCK FORMAT
BFBE- 4A                3120         LSR
BFBF- 05 4A             3130         ORA CONST.AA
BFC1- 9D 8D C0          3140         STA Q6H,X
BFC4- DD 8C C0          3150         CMP Q6L,X
BFC7- 68                3160         PLA
BFC8- C1 00             3170         CMP ($00.X)    DELAY 6 CYCLES
BFCA- 09 AA             3180         ORA #$AA
                        3190 WRITE.BYTE.2
BFCC- EA                3200         NOP
                        3210 WRITE.BYTE.3
BFCD- 48                3220         PHA
BFCE- 68                3230         PLA
BFCF- EA                3240         NOP
BFD0- 9D 8D C0          3250         STA Q6H,X
BFD3- DD 8C C0          3260         CMP Q6L,X
BFD6- 60                3270         RTS
                        3280 *-------------------------------
                        3290 *       VARIOUS ODDS AND ENDS
                        3300 *-------------------------------
BFD7- 01 60             3310         .HS 0160       LEFT OVER
BFD9- 4C DD A5          3320 PATCH1  JMP $A5DD
BFDC- 8D 63 AA          3330 PATCH2  STA $AA63
BFDF- 8D 70 AA          3340         STA $AA70
BFE2- 8D 71 AA          3350         STA $AA71
BFE5- 60                3360         RTS
BFE6- 20 5B A7          3370 PATCH3  JSR $A75B
BFE9- 8C B7 AA          3380         STY $AAB7
BFEC- 60                3390         RTS
BFED- 20 7E AE          3400 PATCH4  JSR $AE7E      FROM $B377
BFF0- AE 9B B3          3410         LDX $B39B
BFF3- 9A                3420         TXS
BFF4- 20 16 A3          3430         JSR $A316
BFF7- BA                3440         TSX
BFF8- 8E 9B B3          3450         STX $B39B
BFFB- A9 09             3460         LDA #9         "DISK FULL" ERROR
BFFD- 4C 85 B3          3470         JMP $B385
```

Commented Listing of DOS 3.3 Format

As promised three or four pages ago, here is my rendition of the DOS 3.3 Format routine.

By the way, there are a lot of differences between DOS 3.2.1 and DOS 3.3 FORMAT routines. Later in this issue of AAL you will find a commented listing of the DOS 3.3 version. If you compare the two, you will find at least these major differences:

1. DOS 3.2.1 formats 13 sectors per track, DOS 3.3 formats 16 sectors per track.
2. DOS 3.2.1 writes an address header followed by a long series of $FF bytes where the data should be; DOS 3.3 writes an address header followed by a standard data block (the data is all $00 bytes).
3. DOS 3.2.1 writes an address header starting with $D5AAB5; DOS 3.3 writes an address header starting with $D5AA96.
4. DOS 3.2.1 verifies correct format by trying to read sector 0 immediately after formatting the last sector; no other verification is made. DOS 3.3 tries to read EVERY sector just formatted; it does a complete check of the track.
5. DOS 3.2.1 writes the sectors in the order 0, 10, 7, 4, 1, 11, 8, 5, 2, 12, 9, 6, 3; DOS 3.3 writes them in sequential order 0, 1, 2, ... , 15.

```
                    1000 *       .LIST OFF
                    1010 *-------------------------------
                    1020 *       DOS 3.3 DISASSEMBLY      $BEAF-BFFF
                    1030 *       BOB SANDER-CEDERLOF        3-26-81
                    1040 *-------------------------------
0578-               1050 RETRY.COUNT          .EQ $578
                    1060 *-------------------------------
C080-               1070 PHASE.OFF            .EQ $C080
C081-               1080 PHASE.ON             .EQ $C081
C088-               1090 MOTOR.OFF            .EQ $C088
C089-               1100 MOTOR.ON             .EQ $C089
C08A-               1110 ENABLE.DRIVE.1       .EQ $C08A
C08B-               1120 ENABLE.DRIVE.2       .EQ $C08B
C08C-               1130 Q6L                  .EQ $C08C
C08D-               1140 Q6H                  .EQ $C08D
C08E-               1150 Q7L                  .EQ $C08E
C08F-               1160 Q7H                  .EQ $C08F
                    1170 *-------------------------------
002D-               1180 SECTOR      .EQ $2D
003E-               1190 CONST.AA    .EQ $3E
003F-               1200 FMT.SECTOR  .EQ $3F
0041-               1210 VOLUME      .EQ $41
0044-               1220 TRACK.CNTR  .EQ $44
0045-               1230 SYNC.CNT    .EQ $45
0048-               1240 IOB.PNTR    .EQ $48,49
                    1250 *-------------------------------
B82A-               1260 WRITE.SECTOR         .EQ $B82A
B8DC-               1270 READ.SECTOR          .EQ $B8DC
B944-               1280 READ.ADDRESS         .EQ $B944
BB00-               1290 RWTS.BUFFER          .EQ $BB00
BC56-               1300 WRITE.ADDRESS        .EQ $BC56
BE5A-               1310 SEEK.TRACK           .EQ $BE5A
BE95-               1320 SETUP.TRACK          .EQ $BE95
                    1330 *-------------------------------
0008-               1340 ERR.CANT.FORMAT      .EQ $08
                    1350 *-------------------------------
                    1360         .OR $BEAF
                    1370         .TA $800
                    1380 *-------------------------------
BEAF- A0 03         1390 FORMAT  LDY #3       POINT AT VOLUME NUMBER
BEB1- B1 48         1400         LDA (IOB.PNTR),Y
BEB3- 85 41         1410         STA VOLUME
BEB5- A9 AA         1420         LDA #$AA     SET UP CONSTANT IN PAGE ZERO
BEB7- 85 3E         1430         STA CONST.AA FOR TIMING
BEB9- A0 56         1440         LDY #86      CLEAR BUFFER TO ALL 00'S
BEBB- A9 00         1450         LDA #0
BEBD- 85 44         1460         STA TRACK.CNTR
BEBF- 99 FF BB      1470 .1      STA RWTS.BUFFER+255,Y
BEC2- 88            1480         DEY          UPPER PORTION
BEC3- D0 FA         1490         BNE .1
BEC5- 99 00 BB      1500 .2      STA RWTS.BUFFER,Y
BEC8- 88            1510         DEY          LOWER PORTION
BEC9- D0 FA         1520         BNE .2
BECB- A9 50         1530         LDA #80      SET UP AS THOUGH IN TRACK 80
BECD- 20 95 BE      1540         JSR SETUP.TRACK
BED0- A9 28         1550         LDA #40      START WITH 40 SYNC'S BTWN SECTORS
BED2- 85 45         1560         STA SYNC.CNT
                    1570 *-------------------------------
BED4- A5 44         1580 .3      LDA TRACK.CNTR
BED6- 20 5A BE      1590         JSR SEEK.TRACK
BED9- 20 0D BF      1600         JSR FORMAT.TRACK
BEDC- A9 08         1610         LDA #ERR.CANT.FORMAT
BEDE- B0 24         1620         BCS .5       ERROR
BEE0- A9 30         1630         LDA #48      TRY UP TO 48 TIMES
BEE2- 8D 78 05      1640         STA RETRY.COUNT
BEE5- 38            1650 .4      SEC
BEE6- CE 78 05      1660         DEC RETRY.COUNT
BEE9- F0 19         1670         BEQ .5       OUT OF RETRIES, ERRCODE=$30
BEEB- 20 44 B9      1680         JSR READ.ADDRESS
BEEE- B0 F5         1690         BCS .4       ERROR, TRY AGAIN
BEF0- A5 2D         1700         LDA SECTOR
BEF2- D0 F1         1710         BNE .4       MUST BE SECOTR 0
BEF4- 20 DC B8      1720         JSR READ.SECTOR
BEF7- B0 EC         1730         BCS .4       ERROR, TRY AGAIN
BEF9- E6 44         1740         INC TRACK.CNTR   NEXT TRACK
BEFB- A5 44         1750         LDA TRACK.CNTR
BEFD- C9 23         1760         CMP #35      FINISHED?
BEFF- 90 D3         1770         BCC .3       NOT YET
BF01- 18            1780         CLC          INDICATE NO ERROR
BF02- 90 05         1790         BCC .6       ...ALWAYS
```

```
                        1800 *-------------------------------
BF04- A0 0D             1810 .5     LDY #13       POINT AT ERROR SLOT IN IOB
BF06- 91 48             1820        STA (IOB.PNTR),Y
BF08- 38                1830        SEC           FLAG ERROR
BF09- BD 88 C0          1840 .6     LDA MOTOR.OFF,X   STOP DRIVE
BF0C- 60                1850        RTS
                        1860 *-------------------------------
                        1870 *      FORMAT A TRACK
                        1880 *-------------------------------
                        1890 FORMAT.TRACK
BF0D- A9 00             1900        LDA #0        START WITH SECTOR 0
BF0F- 85 3F             1910        STA FMT.SECTOR
BF11- A0 80             1920        LDY #128      EXTRA SYNC'S BEFORE FIRST SECTOR
BF13- D0 02             1930        BNE .2        ...ALWAYS
BF15- A4 45             1940 .1     LDY SYNC.CNT
BF17- 20 56 BC          1950 .2     JSR WRITE.ADDRESS
BF1A- B0 6B             1960        BCS .10       ERROR, EXIT NOW
BF1C- 20 2A B8          1970        JSR WRITE.SECTOR
BF1F- B0 66             1980        BCS .10       ERROR, EXIT NOW
BF21- E6 3F             1990        INC FMT.SECTOR  NEXT SECTOR
BF23- A5 3F             2000        LDA FMT.SECTOR
BF25- C9 10             2010        CMP #16       FINISHED WITH THIS TRACK?
BF27- 90 EC             2020        BCC .1        NOT YET
                        2030 *-------------------------------
                        2040 *      VERIFY THE TRACK
                        2050 *-------------------------------
BF29- A0 0F             2060        LDY #15       START WITH SECOIR 15
BF2B- 84 3F             2070        STY FMT.SECTOR
BF2D- A9 30             2080        LDA #48       RETRY UP TO 48 TIMES
BF2F- 8D 78 05          2090        STA RETRY.COUNT
BF32- 99 A8 BF          2100 .3     STA SECTOR.FLAGS,Y  CLEAR ALL THE SECTOR FLAGS
BF35- 88                2110        DEY
BF36- 10 FA             2120        BPL .3
BF38- A4 45             2130        LDY SYNC.CNT DELAY A WHILE
BF3A- 20 87 BF          2140 .4     JSR .10       12 CYCLES
BF3D- 20 87 BF          2150        JSR .10       12 CYCLES
BF40- 20 87 BF          2160        JSR .10       12 CYCLES
BF43- 48                2170        PHA           PHA+PLA=7 CYCLES
BF44- 68                2180        PLA
BF45- EA                2190        NOP           NOP+DEY+BNE=7 CYCLES
BF46- 88                2200        DEY
BF47- D0 F1             2210        BNE .4        WHOLE LOOP = 50 CYCLES
BF49- 20 44 B9          2220        JSR READ.ADDRESS
BF4C- B0 23             2230        BCS .8        ERROR, TRY AGAIN
BF4E- A5 2D             2240        LDA SECTOR    BETTER BE SECTOR 0
BF50- F0 15             2250        BEQ .6        IT IS, HURRAY!
BF52- A9 10             2260        LDA #16       REDUCE # SYNC'S BY TWO
BF54- C5 45             2270        CMP SYNC.CNT UNLESS ALREADY < 16
BF56- A5 45             2280        LDA SYNC.CNT
BF58- E9 01             2290        SBC #1
BF5A- 85 45             2300        STA SYNC.CNT
BF5C- C9 05             2310        CMP #5        IF SYNC.CNT < 5, THERE IS NO HOPE
BF5E- B0 11             2320        BCS .8        >=5, TRY AGAIN
BF60- 38                2330        SEC           FLAG COULDN'T DO IT
BF61- 60                2340        RTS
BF62- 20 44 B9          2350 .5     JSR READ.ADDRESS
BF65- B0 05             2360        BCS .7        ERROR, TRY AGAIN
BF67- 20 DC B8          2370 .6     JSR READ.SECTOR
BF6A- 90 1C             2380        BCC .11       GOOD!
BF6C- CE 78 05          2390 .7     DEC RETRY.COUNT
BF6F- D0 F1             2400        BNE .5        TRY AGAIN
BF71- 20 44 B9          2410 .8     JSR READ.ADDRESS
BF74- B0 0B             2420        BCS .9
BF76- A5 2D             2430        LDA SECTOR
BF78- C9 0F             2440        CMP #15       SECTOR = 15?
BF7A- D0 05             2450        BNE .9        NO
BF7C- 20 DC B8          2460        JSR READ.SECTOR
BF7F- 90 8C             2470        BCC FORMAT.TRACK
BF81- CE 78 05          2480 .9     DEC RETRY.COUNT
BF84- D0 EB             2490        BNE .8        TRY AGAIN
BF86- 38                2500        SEC           FLAG WE COULDN'T DO IT
BF87- 60                2510 .10    RTS           RETURN
                        2520 *-------------------------------
BF88- A4 2D             2530 .11    LDY SECTOR
BF8A- B9 A8 BF          2540        LDA SECTOR.FLAGS,Y
BF8D- 30 DD             2550        BMI .7        ALREADY READ THIS ONE!
BF8F- A9 FF             2560        LDA #$FF
BF91- 99 A8 BF          2570        STA SECTOR.FLAGS,Y
BF94- C6 3F             2580        DEC FMT.SECTOR
BF96- 10 CA             2590        BPL .5
```

```
BF98- A5 44     2600          LDA TRACK.CNTR
BF9A- D0 0A     2610          BNE .12
BF9C- A5 45     2620          LDA SYNC.CNT
BF9E- C9 10     2630          CMP #16
BFA0- 90 E5     2640          BCC .10
BFA2- C6 45     2650          DEC SYNC.CNT
BFA4- C6 45     2660          DEC SYNC.CNT
BFA6- 18        2670 .12      CLC
BFA7- 60        2680          RTS
                2690 *-------------------------------
                2700 SECTOR.FLAGS
BFA8- FF FF FF
BFAB- FF FF FF
BFAE- FF FF     2710          .HS FFFFFFFFFFFFFFFF
BFB0- FF FF FF
BFB3- FF FF FF
BFB6- FF FF     2720          .HS FFFFFFFFFFFFFFFF
                2730 *-------------------------------
                2740 PHYSICAL.SECTOR.VECTOR
BFB8- 00 0D 0B
BFBB- 09 07 05
BFBE- 03 01     2750          .HS 000D0B0907050301
BFC0- 0E 0C 0A
BFC3- 08 06 04
BFC6- 02 0F     2760          .HS 0E0C0A080604020F
                2770 *-------------------------------
                2780 *        CLOBBER WHATEVER IS IN RAM CARD
                2790 *-------------------------------
BFC8- 20 93 FE  2800 PATCH1   JSR $FE93     WHAT PATCH REPLACED
BFCB- AD 81 C0  2810          LDA $C081     WRITE-ENABLE RAM CARD
BFCE- AD 81 C0  2820          LDA $C081
BFD1- A9 00     2830          LDA #0        PUT ZERO IN BYTE WE LATER
BFD3- 8D 00 E0  2840          STA $E000     TEST TO SEE WHICH LANGUAGE
BFD6- 4C 44 B7  2850          JMP $B744     RETURN
                2860 *-------------------------------
                2870 *-------------------------------
                2880 *        VARIOUS ODDS AND ENDS
                2890 *-------------------------------
BFD9- 00 00 00  2900          .HS 000000
BFDC- 8D 63 AA  2910 PATCH2   STA $AA63
BFDF- 8D 70 AA  2920          STA $AA70
BFE2- 8D 71 AA  2930          STA $AA71
BFE5- 60        2940          RTS
BFE6- 20 5B A7  2950 PATCH3   JSR $A75B
BFE9- 8C B7 AA  2960          STY $AAB7
BFEC- 60        2970          RTS
BFED- 20 7E AE  2980 PATCH4   JSR $AE7E     FROM $B377
BFF0- AE 9B B3  2990          LDX $B39B
BFF3- 9A        3000          TXS
BFF4- 20 16 A3  3010          JSR $A316
BFF7- BA        3020          TSX
BFF8- 8E 9B B3  3030          STX $B39B
BFFB- A9 09     3040          LDA #9        "DISK FULL" ERROR
BFFD- 4C 85 B3  3050          JMP $B385
```

Substring Search Function for Applesoft

Lee Reynolds' article in the January 1981 Call A.P.P.L.E. touched off this project. When you are searching through text arrays for keywords, or through a mailing list for someone who lives on "XYZ Street", Applesoft can be vveeerrrrryyy slow. This subroutine, linked in through the famous ampersand feature, will give you the speed your Apple is famous for.

Lee's program was quite similar to this one, but it did not allow the keyword or the string-to-be-searched to be expressions. He left that extension as "an exercise for the reader". Being one reader badly in need of exercise, I took up the challenge.

Although it is not really necessary, I used one of the newly discovered "secret" opcodes (which I wrote about last month) at line 2060. If you like, you can replace that line with:

```
2060 GS1    LDA (FACMO),Y
2065        TAX
```

```
                1010 *--------------------------------
                1020 *
                1030 *      SUBSTRING SEARCH FUNCTION FOR APPLESOFT
                1040 *      ---------------------------------------
                1050 *
                1060 *      & SUB$( A$, B$, I )
                1070 *
                1080 *      SEARCHES FOR FIRST OCCURRENCE OF
                1090 *      B$ IN A$; PUTS RESULT IN I
                1100 *
                1110 *      RETURNS I=0 IF B$ IS NOT IN A$
                1120 *
                1130 *      (REFERENCE:  CALL A.P.P.L.E. ARTICLE
                1140 *      IN JANUARY 1981 ISSUE BY LEE REYNOLDS,
                1150 *      PAGES 26-30.)
                1160 *
                1170 *--------------------------------
00A0-           1180 FACMO        .EQ $A0
0052-           1190 TEMPPT       .EQ $52
0018-           1200 MAIN.LENGTH  .EQ $18
0019-           1210 MAIN         .EQ $19,1A
001B-           1220 KEY.LENGTH   .EQ $1B
001C-           1230 KEY          .EQ $1C,1D
                1240 *--------------------------------
DA5C-           1250 ASSIGN .EQ $DA5C     STORE VALUE IN VARIABLE
DEC0-           1260 SYNCHR .EQ $DEC0     REQUIRE (A) AS NEXT CHAR
DD7B-           1270 FRMEVL .EQ $DD7B     EVALUATE FORMULA
DEBE-           1280 SYNCOM .EQ $DEBE     REQUIRE COMMA
DEB8-           1290 SYNRPN .EQ $DEB8     REQUIRE ")"
DD6C-           1300 CHKSTR .EQ $DD6C     REQUIRE STRING
DFE3-           1310 PTRGET .EQ $DFE3     GET POINTER
E604-           1320 FRETMP .EQ $E604     FREE TEMPORARY STRING
E301-           1330 SNGFLT .EQ $E301     FLOAT (Y)
                1340 *--------------------------------
                1350        .OR $300
                1360        .TF B.SUBSTRING SEARCH
```

```
                    1370 *--------------------------------
                    1380 SETUP.AMPERSAND
0300- A9 4C         1390         LDA #$4C      JMP OPCODE
0302- 8D F5 03      1400         STA $3F5
0305- A9 15         1410         LDA #SUB
0307- 8D F6 03      1420         STA $3F6
030A- A9 03         1430         LDA /SUB
030C- 8D F7 03      1440         STA $3F7
030F- 60            1450         RTS
                    1460 *--------------------------------
0310- 28 24 42
0313- 55 53         1470 SUBQT   .AS "($BUS"   SUB$( BACKWARDS
                    1480 *--------------------------------
                    1490 SUB
0315- A2 04         1500         LDX #4        COMPARE FOR "SUB$("
0317- BD 10 03      1510 .1      LDA SUBQT,X
031A- 20 C0 DE      1520         JSR SYNCHR    COMPARE WITH INPUT
031D- CA            1530         DEX
031E- 10 F7         1540         BPL .1
                    1550 *--------------------------------
0320- A0 18         1560         LDY #MAIN.LENGTH
0322- 20 69 03      1570         JSR GET.STRING
0325- A0 1B         1580         LDY #KEY.LENGTH
0327- 20 69 03      1590         JSR GET.STRING
032A- 20 E3 DF      1600         JSR PTRGET    GET VARIABLE FOR RESULT
032D- 85 85         1610         STA $85
032F- 84 86         1620         STY $86
0331- 20 B8 DE      1630         JSR SYNRPN    REQUIRE RIGHT PAREN
                    1640 *--------------------------------
0334- 20 86 03      1650         JSR FREE.STRINGS
                    1660 *--------------------------------
0337- A2 00         1670         LDX #0        ANSWER OFFSET
0339- A5 18         1680 .2      LDA MAIN.LENGTH SEE IF IT CAN STILL FIT
033B- C5 1B         1690         CMP KEY.LENGTH
033D- 90 26         1700         BCC .8        WILL NOT FIT
033F- A0 00         1710         LDY #0
0341- B1 1C         1720 .3      LDA (KEY),Y
0343- D1 19         1730         CMP (MAIN),Y
0345- D0 13         1740         BNE .6
0347- C8            1750         INY
0348- C4 1B         1760         CPY KEY.LENGTH
034A- 90 F5         1770         BCC .3
034C- E8            1780         INX           X IS RESULT
034D- 8A            1790         TXA
034E- A8            1800         TAY
034F- 20 01 E3      1810 .4      JSR SNGFLT    FLOAT THE BYTE IN Y
0352- A5 12         1820         LDA $12
0354- 48            1830         PHA
0355- A5 11         1840         LDA $11
0357- 4C 5C DA      1850         JMP ASSIGN    STORE VALUE IN VARIABLE
035A- E6 19         1860 .6      INC MAIN
035C- D0 02         1870         BNE .7
035E- E6 1A         1880         INC MAIN+1
0360- E8            1890 .7      INX
0361- C6 18         1900         DEC MAIN.LENGTH
0363- D0 D4         1910         BNE .2
0365- A0 00         1920 .8      LDY #0        RESULT IS 0
0367- F0 E6         1930         BEQ .4        ...ALWAYS
```

```
                    1940 *--------------------------------
                    1950 *      GET STRING EXPRESSION
                    1960 *--------------------------------
                    1970 GET.STRING
0369- 8C 7A 03      1980         STY GS2       PLUG OUTPUT VECTOR
036C- 20 7B DD      1990         JSR FRMEVL    EVALUATE FORMULA
036F- 20 BE DE      2000         JSR SYNCOM    REQUIRE TRAILING COMMA
0372- 20 6C DD      2010         JSR CHKSTR    REQUIRE STRING
0375- A0 02         2020         LDY #2        GET STRING DATA
                    2030 * THE NEXT LINE IS A "SECRET" 6502 OPCODE,
                    2040 * WHICH DOES BOTH LDA (FACMO),Y AND LDX (FACMO),Y
                    2050 * AT THE SAME TIME.
0377- B3 A0         2060 GS1     .DA #$B3,#FACMO
0379- 96 00         2070         STX *-*,Y     PLUGGED IN FROM ABOVE
037A-               2080 GS2     .EQ *-1
037B- 88            2090         DEY
037C- 10 F9         2100         BPL GS1
037E- 60            2110         RTS
                    2120 *--------------------------------
                    2130 *      FREE UP ANY TEMPORARY STRINGS
                    2140 *--------------------------------
                    2150 FREE.ONE.STRING
037F- A5 53         2160         LDA TEMPPT+1
0381- A0 00         2170         LDY #0
0383- 20 04 E6      2180         JSR FRETMP
                    2190 FREE.STRINGS
0386- A5 52         2200         LDA TEMPPT
0388- C9 56         2210         CMP #$56      EMPTY?
038A- B0 F3         2220         BCS FREE.ONE.STRING
038C- 60            2230         RTS
```

Here is a sample Applesoft program which uses the Substring Search Subroutine. Line 10 loads the subroutine and calls 768 to link in the ampersand vector. Line 120 reads in your search key. If you just hit the RETURN key, the program quits.

Line 130 gets the next string to be searched from the DATA list. If the value is ".", we are at the end of the list, so it loops back to line 110.

Line 140 calls our substring search subroutine to see if the key string can be found in the search string. If not, it jumps back to line 130 to get another search string. Lines 150-180 print the search string, emphasizing the portion that matched the key string by printing it in inverse.

```
10  PRINT  CHR$ (4)"BLOAD B.SUBSTRING SEARCH": CALL 768
100  DATA ASM,DELETE,FAST,FIND,HIDE,INCREMENT,LIST,LOAD,MEMORY,ME
     RGE,MGO,NEW,PRT,RENUMBER,RESTORE,SAVE,SLOW,USER,VAL,.
110  RESTORE
120  INPUT "KEY STRING: ";K$: IF K$ = "" THEN  END
130  READ A$: IF A$ = "." THEN  PRINT : GOTO 110
140  & SUB$(A$,K$,I): IF I = 0 THEN 130
150  IF I > 1 THEN  PRINT  LEFT$ (A$,I - 1);
160  INVERSE : PRINT K$;: NORMAL
170 L =  LEN (A$) - I + 1 -  LEN (K$): IF L > 0 THEN  PRINT  RIGHT$
     (A$,L);
180  PRINT : GOTO 130
```

Apple Assembly Line is published monthly by S-C SOFTWARE, P. O. Box 5537, Richardson, TX 75080. Subscription rate is $12/year, in the U.S.A., Canada, and Mexico. Other countries add $6/year for extra postage.