WASHINGTON APPLE PI

Vol. 1 No. 3
April, 1979

Dear APPLE Enthusiast,

Our thanks again to John Moon for a job well done on the Constitution and By-Laws. The review session at the last meeting was long, but necessary, and we now have a constitution (and are all the better for it). To save postage costs, the revised version will not be sent with this newsletter but will be available as a handcut at our next meeting. Those of you who won't be able to make the next meeting can get a copy by sending me a self-addressed stamped envelope.

It was gratifying to me to see the turnout of 38 people at our last meeting. We must be doing something right. Word that we exist is indeed spreading, and that we are evolving into a professional, useful and fun group with something of interest for all APPLE users, including our green apples.

I have both good news and "bad" news for you. First the good:

1. Tom Bottegal of GWU says that he and his crew are willing to operate an APPLE II HOTLINE (676-6 853) and to attempt to answer questions on what you do after you hit reset--or whatever. If they or others in the group can't field the questions, we can then pass them on to APPLI.. A free newsletter to the first person who stumps the HOTLINERS!

2. Tom also agrees to allow group members access to the GWU library and its collection of many of the personal computer magazines. Xeroxing is also available at 5¢ per page.

3. The eight API'LES at GWU are available for show and tell, program exchange, etc. after we finish our formal meeting If you feel the need to bring your own computer because you don't like to pul. out your Superchip or whatever, you can do that too.

Now the "bad":

1. Not all of you are doing what you ought--we need your inputs to this newsletter. Come on, send them in whether or not they are written in pearly prose. We'll fix the language, if necessary. This is your newsletter--and this includes all you young people too! Rick Hodder needs items for his Green Apples column.

2. This next item may prove to be unpopular, but I think it requires airing and some thought by all members. I believe that a(if not the) major benefit that

accrues to the membership is the sharing of programs written by them or obtained from others who are also willing to share. However, I believe that the group should not sanction the free and extensive exchange of programs that have been prepared and marketed or copyrighted by others. I believe that this exchange will continue to occur but I recommend that the group curtail this practice during our meetings at GWU. I will bring this up at our next session to hear your views.

Bernard Urban

Look out PLATO - Here comes the APPLE!

PLATO, the multi-million dollar program funded by NSF and used by the University of Illinois, University of Delaware and elsewhere for computer assisted instruction in the classroom, now has a competitor--our own APPLE. Ron Thorkildsen at the Exceptional Child Center of the University of Utah has interfaced a video disc and touch panel to the APPLE. He will be giving a paper on May 17 at the Pentagon Motel in Arlington about its use to tutor learning disabled children.

Minutes of the 3/31/79 Meeting

The major portion of the meeting was devoted to review, modification and adoption of the Constitution and By-Laws, with John Moon presiding.

Hal Weinstock described what is available and what can be arranged in the way of courses for a fee, for the APPLE II. While not turning off these possibilities, the members once again stated that there is probably sufficient expertise within our group to provide instruction on IBASIC, Monitor, Sweet Sixteen, HIRES, DOS, APPLESOFT II, etc. Classes may be held at GWU after the regular meetings or perhaps weekly.

Bernie Urban was voted in as President until the May elections. He designated John Moon as Vice-President, Mark Crosby as Program Chairman, John Ditman as Treasurer, and Genevie Urban as Secretary.

No definitive action was taken on establishing a dues structure. Attendees were requested to contribute $1.00 to offset the existing deficit and to provide postage for this issue. The group agreed that Bernie should pursue the possibilities for incorporation of all local APPLE users' groups into one in order to gain not-for-profit status and 3rd class mail privileges. Local chapters or divisions would still exist with their own officers and operations, but there would be one newsletter with contributions from all. This could result in a high quality, useful publication.

--From the "Green Apples"--

Congratulations to Andy Rose on his game "SHOOTOUT" (a fine LORES game written by two 11-year olds), which has been accepted by APPLE's Software Bank. As a result of this he received copies of Software Bank's Volumes III, IV and V, which he is willing to share with our group. The programs will be on diskettes for copying at Computers, Etc. and at our next meeting.

SOFTWARE REVIEW by Mark Crosby

Many of you bought the APPLE II because of its graphic capabilities. After running the hi-resolution demonstration several hundred times and after trying to write your own, you might have given up and purchased Starwars or Lunar Lander or perhaps you have gone on to arcade-quality games like Bomber.

Well, I have just purchased a simulation "shoot-em-up" based on Starwars but it bears no resemblance to the version you probably have seen and tried.

SUPER STARWARS by Programma International, Inc., is a real-time simulation of the battle scenes in the movie Starwars. Let me describe how it works: After giving you instructions and offering three modes of play 1) Auto-Fire with joystick, 2) Manual Fire with joystick, 3) Remote controlled torpedoes with joystick (yes, remote controlled!), and setting the level (0-9), the game commences with a beautiful simulation of the moment the good guys come out of hyperspace travel (light speed). All the stars just sort of stretch in a Z-O-O-O-O-O-M pattern complete with terrific sound effects.

Then the stars solidify and, from the central area of the screen, appear small craft (the "enemy") which get bigger and bigger and head slight off-center toward your ship. You are in your ship and see the view-screen and a cross-hair for aiming your laser weapon. You control both the velocity and the direction of movement of the cross-hair using a joystick. When you hit the "enemy", there is a sound and an explosion at the exact point at which you hit the "enemy" ship. The explosion gets larger as it dissipates and the ship vanishes. In the remote-controlled mode, you release a torpedo, send it away from you and it curves off in the direction the joystick is pointing.

The "enemy" will fire lasers at you as they approach and this becomes more frequent as they get closer to you. The screen flashes white with each blast and you lose some power. During moments of rest (micro-seconds, really), your power builds back up. When you have lost enough power, it's "curtains" for you and your screen flashes and rolls with a nasty sound. Conect your APPLE to an external speaker for this one!

There is also overlap - sometimes there are several "enemy" ships appearing depending on the skill level you have chosen and the mode. When you hit one of them, the others keep right on coming during the explosion of the first (there is no "lock-up" of the joystick movements). If "enemy" ships are near to each other and the same apparent distance from you, an explosion on one can trigger an explosion on the other.

Make no mistake - this is not a run-of-the mill program. The simulation is most complete, three-dimensional, full of good sounds, real-time and a real challenge to even the most seasoned gamester. By the way, there is no grading of your score - you either make it to fight another day or you get wiped out!

SUPER STARWARS - loads from $800-$6000 (24K) $15.95

from: Programma International, Inc.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579/1116/1117

*** A REAL BALL-BOUNCER!*** by Howie Mitchell

(Inspired by Mark Crosby's Lissajous program!)

Basic Program outline: (Applesoft II "up")

```
10   TEXT : HOME : VTAB 3 : HTAB 5 : PRINT "*** A REAL
     BALL-BOUNCER! ***": PRINT
20   HTAB 9: PRINT "BY: HOWIE MITCHELL": PRINT : HTAB 9 :
     PRINT "APRIL 1, 1979": PRINT
30   FOR BOUNCE = 1 TO 10: FOR BEEP = 0 TO 1000 - 80* BOUNCE:
     NEXT BEEP: PRINT " ctrl.G": NEXT BOUNCE
40   VTAB 10: PRINT "  THIS PROGRAM WILL PLOT THE TRAJECTORY
     OF A BALL, DROPPED (OR THROWN!) INTO A LARGE ROOM.": PRINT
50   PRINT "  THE BALL ENTERS THROUGH THE CEILING, TRAVELS
     STRAIGHT DOWN, AND BOUNCES OFF A BOARD (WHOSE ANGLE OF TILT
     MAY BE"
60   PRINT "SPECIFIED). THE BALL'S INITIAL SPEED, AND ITS
     'BOUNCINESS' MAY ALSO BE ENTERED. THE RESULT IS AN ENDLESS
     VARIETY OF BOUNCE PATHWAYS !"
70   GOSUB 2000
80   INPUT "BALL'S SPEED OF ENTRY ? ";VV: PRINT
90   INPUT "BOARD'S ANGLE OF TILT ? ";BA: PRINT
100  INPUT "BALL'S BOUNCINESS ? (USE DECIMAL FROM 0 TO 1) "; BR
110  HGR (or:HGR2) : HCOLOR= 3: S= -16336
120  FOR X = 10 TO 270: HPLOT X,0: NEXT X: FOR Y = 0 TO 180:
     HPLOT 270,Y: NEXT Y: FOR X = 270 TO 0 STEP -1: HPLOT X,180:
     NEXT X: FOR Y = 180 TO 0 STEP -1: HPLOT 0,Y: NEXT Y
125  PRINT "ctrl.G"
130  HPLOT 0,170 TO 10,180
135  Y = 0
140  DT = .075: VV = VV + 32*DT
150  Y = Y + VV * DT
160  IF Y> 180 THEN GOTO 200
170  HPLOT 5,Y: GOTO 140
200  ANGLE = BA*2/57: VH = VV* SIN(ANGLE): VV = -VV* COS(ANGLE)
205  FOR Z = 0 TO 10: SOUND = PEEK(S)+PEEK(S)+PEEK(S): NEXT Z
210  X = 5: Y = 175
220  X = X + VH * DT: VV = VV + 32 * DT: Y = Y + VV * DT
230  IF X> 270 OR X< 0 THEN GOSUB 500
235  IF Y> 180 OR Y< 0 THEN GOSUB 520
240  HPLOT X,Y: GOTO 220
500  IF X> 270 THEN X = 270
505  IF X< 0 THEN X = 0
510  VH = - VH * SQR (BR)
515  FOR SOUND = 1 TO 5: B= PEEK(S) + PEEK(S): NEXT SOUND: RETURN
520  IF Y> 180 THEN Y = 180
525  IF Y< 0 THEN Y = 0
530  VV = - VV * SQR (BR)
535  IF Y = 180 AND ABS (VV)< 1.5 THEN HPLOT X,160 TO X,175 : STOP
540  SOUND = PEEK (S) + PEEK (S) + PEEK (S): RETURN
2000 VTAB 23: INPUT "  (PRESS 'RETURN' TO CONTINUE.) "; HOLD$:
     PRINT : PRINT : RETURN
```

Some notes on the Ball-bounce Program:

LINE #30: I discovered by accident that the bell sound may be included in a program by simply making the ctrl.G function part of a print statement. It is NOT visible in the program printout (one sees PRINT ""), nor does it copy by running the cursor over it via the right-moving arrow key.

LINE #100: Here, a bounciness of greater than 1 may be used, with a most surprising result !

LINE #150: This is a variation on the freefall equation $Y = V_0T + \frac{1}{2} gT^2$, showing a slight influence of the Calculus. Y = Y + VV*DT runs much faster.

LINE #200: Converts bounce angle (BA) into radians, and produces horizontal and vertical speed components for a bounce from the tilted board.

LINE #510: Here, using the square root of BR is for some reason necessary for causing the "ball" to bounce back to BR * its previous bounce height (e.g. If BR = .9, then each new bounce height is about 9/10 of the previous height.). The handsome "envelope" of the bounces can be seen nicely by using ENTRY SPEED = 0, TILT ANGLE = .75, and BOUNCINESS = .91..

CHR$ for Integer Basic by Jim Rose

For some mystifying reason the WOZ built the string-to-integer function ASC into Integer Basic but omitted the companion function CHR$ which translates an integer back into the string representation of the byte. There are some expensive solutions to this problem: use Applesoft which indeed does have this function (and either gobbles a substantial part of my core, or my budget for the card); or do without, program around, and otherwise get frustrated.

A less expensive solution to this problem is also quite simple, and substantially expands the string processing capabilities of Integer Basic. Try this:

```
 0 DIM V$(1)
10 FOR I= 160 TO 223
20   POKE 2053,I
30   PRINT I; "="; V$,
40 NEXT I: END
```

In effect, by dimensioning the character string V$ first (in line 0), I know the memory byte location of the character (2053). I have thus set up the recursive relationship:

V$= POKE 2053, ASC(V$)

which is exactly parallel to the functional recursion:

V$= CHR$(ASC(V$))

that I'm trying to emulate.

There are three restrictions on the use of this little trick. The string variable (V$) must be a single letter variable (A$ through Z$). Otherwise the byte location (2053) must be changed.* This resident string variable must be dimensioned first, or again the byte location is changed. And the third constraint is that the value which you POKE must be less than 256, and in normal use should range from 160 to 223.

My search for a CHR$ function is actually part of my need for dimensioned string arrays. Again I could use Applesoft, but with only 16K my capabilities then to do anything are severely restricted. With this Integer Basic patch I am off and running with all my memory available for use.

The following program will give you a simple idea of how this works. Line 0 dimensions my CHR$, and sets up my string array with 100 records of 40 characters each. Lines 100-130 allow me to input records (N positive), or display records already stored (N negative). Lines 1000-1040 fill my storage array, and lines 2000-2060 retrieve the data.

```
  0 DIM V$(1), A$(40), D(4000)
100 INPUT N
110 IF N = 999 THEN 9000
120 GOSUB 1010 + 1000 * (N# ABS(N))
130 GO TO 100

1000 REM   SUBROUTINE PUT
1001 REM   THIS STORES THE N'TH RECORD IN D
1010   N = 40 * N : INPUT A$
1020   FOR I = 1 TO LEN( A$)
1030      D( I + N) = ASC( A$( I,I))
1040   NEXT I : RETURN

2000 REM   SUBROUTINE GET
2001 REM   THIS RETRIEVES THE N'TH RECORD FROM D
2010   N = -40 * N : A$ = ""
2020   FOR I = 1 TO 40
2025      IF D(I+N) > 223 OR D(I+N) < 160 THEN 2060
2030      POKE 2053, D(I+N)
2040      A$(I) = V$
2050   NEXT I
2060   PRINT A$ : RETURN

9000 END
```

*Actually the byte location = 2052 + (the number of characters in the string name)

One more note. In this simple program I am actually using twice as much storage as I need. The array D(4000) is 4001 words long, or 8002 bytes! Since each string character takes only one byte, I can pack two characters for every word of D, and thus double the storage capacity. I'll leave this patch as an exercise for the reader.

A CALL/PEEK/POKE LISTING (PARTIAL)

DESCRIPTION	IN BASIC, USE	IN MACHINE LANG.	COMMENTS
Enter Basic	(-8192)	JMP $EØØØ	*EØØØG
Ring Bell	CALL-198	JSR $FF3A	CNTRL-G IN " "
RING BELL & PRINT ERR	CALL-211	JSR $FF2D	ACC,Y
CASSETTE OUT TOGGLE TIME	POKE-16352,Ø	STA $CØ2Ø	
CLEAR KEYBOARD STROBE	POKE-16368,Ø	STA $CØ1Ø	
CLEAR SCREEN,HOME CURSOR	CALL-936	JSR $FC58	ACC,Y
CLEAR SCREEN,CURSOR TOEOP	CALL-958	JSR $FC42	ACC,Y
CLEAR SCREEN,CURSOR TOEOL	CALL-868	JSR $FC9C	ACC,Y
CLR(CLEAR VARIABLES)	CALL-6729	JSR $E5B7	
SET COLOR	COLOR=XX	JSR $F864,LDA #COLOR	(-1948)
INCREMENT COLOR BY (3)	CALL-1953	JSR $F85F	
INCREMENT COLOR BY (1)	SEE COMMENTS	INC $3Ø	POKE 48,PEEK(48)+1
CON(CONTINUE)	CALL-3318	JMP $F3ØA	F3ØAG
READ CURSOR (HORIZ)	X=PEEK(36)	LDA $24	
SET CURSOR (HORIZ)	POKE36,X	STA $24	
READ CURSOR(VERTICAL)	X=PEEK(37)	LDA $25	
SET CURSOR (VERTICAL)	POKE37,X	STA $25	
SET FLAG(NORMAL)	POKE5Ø,255	LDA #FF STA $32	
SET FLAG(INVERSE)	POKE5Ø,63	LDA #3F STA $32	CALL-384 IN BASIC
SET FLAG(FLASHING)	POKE5Ø,127	LDA #CF STA $32	
GAME I/O SET (ANØ)	POKE-16295,Ø	STA $CØ59	
GAME I/O SET (AN1)	POKE-16293,Ø	STA $CØ5B	
GAME I/O SET (AN2)	POKE-16291,Ø	STA $CØ5D	
GAME I/O SET (AN3)	POKE-16289,Ø	STA $CØ5F	
GAME I/O CLEAR (ANØ)	POKE-16296,Ø	STA $CØ58	
GAME I/O CLEAR (AN1)	POKE-16294,Ø	STA $CØ5A	
GAME I/O CLEAR (AN2)	POKE-16292,Ø	STA $CØ5C	
GAME I/O CLEAR (AN3)	POKE-1629Ø,Ø	STA $CØ5E	
GR,SET GRAPHICS MODE	CALL-1216	JSR $FB4Ø	
GR,SET GRAPHICS MODE	POKE-16304,Ø	STA $CØ5Ø	
IN GR MODE (CLEAR SCREEN)	CALL-1998	JSR $F832	ACC,Y
CLEAR TO TEXT	POKE-16303,Ø	STA $CØ51	
CLEAR TO TEXT	CALL-1223	JSR $FB39	ACC
SET MIXED GR/TEXT	POKE-16301,Ø	STA $CØ53	
CLEAR MIXED MODE(ALL GR)	POKE-16302,Ø	STA $CØ52	
SET HI-RES MODE	POKE-16297,Ø	STA $CØ57	
CLEAR HI-RES MODE	POKE-16298,Ø	STA $CØ56	
SET PAGE 2	POKE-16299,Ø	STA $CØ55	EITHER LOW OR HI-RES
SET PAGE 1	POKE-163ØØ,Ø	STA $CØ54	OR TEXT
HEX,PRINT (1 DIGIT #)	CALL-541	LDA #XX, JSR $FDE3	
HEX,PRINT (2 DIGIT #)	CALL55Ø	LDA #XX, JSR $FDDA	
HEX,PRINT (4 DIGIT #)	C-1728)	LDY #1ST 2-DIGITS	
		LDX #2ND 2-DIGITS	
HLIN	HLIN A,B AT C	LDA #B	B
		STA $2C	
		LDA #C	C
		STA #AC	$PC OR ACC
		LDY #A	A
	(-2Ø23)	JSR $F819	
INPUT(DIRECT)	CALL-528	JSR $FDFØ	1 CHAR
INPUT (FULL LINE) TO RET	CALL-662	JSR $FD6A	
INPUT (GET CHAR FROM KBD)	CALL-715	JSR $FD35	ACC,Y
INPUT (READ KBD STROBE)	X=PEEK(-16384)	LDA $CØØØ	
INPUT (CLEAR KBD STROBE)	POKE-16368,Ø	STA $CØ1Ø	
LINE FEED	CALL-922	JSR $FC66	
LIST COMPLETE PROGRAM	CALL-8117	JSR $EØ4B	
LIST LIN.# TO LINE#	LIST 1Ø,5Ø	LDA #LOL,LOL=LO BYTE	LAST MEM LOC
		STA $E2	
		LDA #HIL,HIL=HI BYTE	OF LOC
		STA $E3	
		LDA #LOF,LOF=LO BYTE	OF LAST LOC
		STA $E6	
		LDA #HIF,HIF=HI BYTE	OF LOF
		STA $E7	
USE LOC OF LINE# NOT LINE#	(-8133)	JSR $EØ3B	
LOAD (BASIC PROG)	CALL 3873	JSR $FØDF	
MAN	CALL-4524	JSR $EE54 OR EE54G Ø	
MOVE (MEM TO MEM)		*(NEW START) (OLD START).(OLD END)M	
		*1ØØØ<8ØØ.8ADM	FROM MONITOR
NEW	CALL-6739	JSR$E5AD	
NO TRACE	CALL-3722	JSR $F176	

DESCRIPTION	IN BASIC, USE	IN MACHINE LANG.	COMMENTS
OUTPUT (ROUTINE)	CALL-739	LDA #ASCII,ASCII=	DESIRED CHAR
PDL(READ)	X=PDL(#)	LDX #PDL,PDL=PDL#Ø-3	ACC,Y
PDL (SWITCH (0) READ)	X=PEEK(-16287)	LDA $CØ61	
PDL (SWITCH (1) READ)	X=PEEK(-16286)	LDA $CØ62	
PDL (SWITCH (2) READ)	X=PEEK(-16285)	LDA $CØ63	
PLOT	PLOT X,Y	LDX #X	
		LDY #Y	
	CALL-2Ø48	JSR $F8ØØ	
PRINT (DUMMY)	PRINT,CALL6115	JSR $E81D	
REGISTERS USE PAGE-Ø LOCN	CALL-1321	JSR $FAD7,SCROLLS-1	LOC $46 FOR X REG
REGISTERS (OPEN)	CALL-1312	JSR $FAEØ,NO SCROLL	LOC $47 FOR Y REG
REGISTERS (OPEN)		JSR $FADE	LOC $48 FOR STATUS RI
REGISTERS (OPEN)			LOC $49 FOR STACK PO
REGISTERS (RESTORE)	CALL-193	JSR $FF3F	
REGISTERS (SAVE)	CALL-182	JSR $FF4A	
RESET TO MONITOR	CALL-167	JSR $FF59	
ENTER WITHOUT BELL	CALL-155	JSR $FF65	
ENTER (SOFT)	CALL-151	JSR $FF69	PRESERVES DOS,ETC
RND (FUNCTION)	X=RND(#)	LDA #LO,LO=LO BYTE OF	HIGHEST # DESIRED
		STA $CE	
		LDA #HI,HI=HI BYTE OF	HIGHEST # DESIRED
		STA $CF	
		LDX #2Ø	
	CALL-4271	JSR $EF51	
RUN (CLR VARIABLES)	CALL-4116	JMP $EFEC	
RUN (SAVE VARIABLES)	CALL-6Ø9Ø	JMP $E836	
RUN USING CTRL-Y		LDA #LO,LO=LO BYTE OF	LOCN,NOT LINE #
		LDY #HI,HI=HI BYTE OF	LOCN,NOT LINE #
	CALL-6Ø86	JMP $E83A	
AT LOCN $Ø3F8 INSTALL A	JSR $LOCN	TO RUN M/L PROG	
SAVE	CALL-3774	JSR $F142	
SCRN		LDY #X,X=X COORDINATE	
		LDA #Y,Y=Y COORDINATE	
ACC RETURNS WITH SCRN AT	X,Y CALL-1935		
SCROLL(1 SPACE)	CALL-912	JSR $FC7Ø	ACC,Y
SPEAKER (TOGGLE)	POKE-16336,Ø	STA $CØ3Ø	
SWEET 16	CALL-3959	$F689	
TEXT (SET WINDOW TO WIDEST	POKE-163Ø3,Ø	STA $CØ51	
TEXT,(OPEN WINDOW TO WIDST	TEXT	JSR $FB2F	
TEXT,(SET TEXT MODE)	TEXT	JSR $3B39	
TRACE	CALL-3727	JSR $F171	
VLIN	VLIN A,B AT C	INFO NOT AVAIL AT THIS TIME	
VTAB	VTAB #	LDA #ROW	ACC
		JSR $FC22	(-99Ø)
WAIT	CALL-856	JSR $FCA8	
WINDOW (SET LEFT)	POKE32,LEFT	LDA #LEFT	
		STA $2Ø	
WINDOW (SET WIDTH)	POKE33,WIDTH	LDA #WIDTH	
		STA $21	
WINDOW (SET TOP)	POKE34,TOP	LDA #TOP	
		STA $22	
WINDOW (SET BOTTOM)	POKE35,BOTTOM	LDA #BOTTOM	
		STA $23	
WRITE (TO TAPE)		NOT AVAIL AT THIS TIME	

$FE8Ø: SETS INVERSE VIDEO
$FE84: SETS NORMAL VIDEO
$FE89: SETS KEYBOARD AS AN INPUT DEVICE
$FE93: SETS CRT AS OUTPUT DEVICE
$FE8B: SETS AN INPUT DEVICE TO SLOT SPECIFIED BY ACC
$FE95: SETS OUTPUT DEVICE TO SLOT SPECIFIED BY ACC
$CC,CD POINTS TO START OF PROGRAM, $CA,CB POINTS TO END OF VARIABLES TABLE
$4C,4D IS HIMEM

GATHERED AND RESEARCHED BY ED AVALAR TAKEN FROM ABACUS FEBRUARY 1979 VOLUME #1 ISSUE #2 AND EDITED BY MARK CROSBY OF WASHINGTON APPLE PI WITH PERMISSION.

BASIC TO MACHINE LANGUAGE ROUTINE INTERFACING by John L. Moon

The other day I needed to print out a formatted dump of a data area. I was working on a P-code interpreter and wanted a hexadecimal dump as part of my debug package. Also, I wanted a way to use machine language routines in the Monitor that required values to be passed in registers. The following BASIC to machine language routines are what resulted from these efforts.

In the APPLE Monitor at lacation F940 is a 6502 routine that takes the contents of the Y and X registers and prints them out as a four-digit hexadecimal number. Unfortunately, the CALL interface to machine language from BASIC has no capability to load any of the processor registers. However, there are several machine language instructions that can load or store the 6502 registers. Therefore, a machine language routine can be written that when called from BASIC can load the registers with the desired values. In this case, the routine could look like this:

```
LDY # byte 1        Load Y register with byte 1
LDX # byte 2        Load X register with byte 2
JSR $F940           Call the hex print routine
RTS                 Return to BASIC
```

For convenience, the routine can be loaded into locations starting at machine address 0 by the following lines of BASIC code (I've hand assembled the program and converted it to decimal for the POKES):

```
10 POKE 0,160: POKE 2,162: POKE 4,32: POKE 5,64: POKE 6,249:
   POKE 7,96
```

In order to use the routine from BASIC the following BASIC subroutine should be used to put the data into the instructions so that it will be loaded into the proper registers:

```
1000 REM  HB IS HIGH BYTE, LB IS LOWBYTE
1001 POKE 1,HB: POKE 3,LB: CALL 0:RETURN
```

An example of a calling sequence could be:

```
100 INPUT "WHICH ADDRESS", ADDR
110 HB = PEEK(ADDR): LB = PEEK(ADDR+1): GOSUB 1000: GOTO 100
```

This program works by POKEing into the 6502 LDY and LDX instructions the data byte that is to be loaded into the appropriate register.

With a little extension to the machine language interface routine, a general purpose BASIC to machine language interface can be created that is capable of loading or returning values in any of the 6502 registers and is capable of calling any machine language routine. The machine language portion of this routine is:

Location	Mnemonic	Operand	Comments
0	LDA	#aregbyte	Loads the A register
2	LDX	#xregbyte	Loads the X register
4	LDY	#yregbyte	Loads the Y register
6	JSR	$machineaddr	Calls the 6502 routine
9	STA	$01	Stores A register at loc 1
11	STX	$03	Stores X register at loc 3
13	STY	$05	Stores Y register at loc 5
15	RTS		Returns to BASIC

This routine can be loaded into memory from BASIC with the following POKEs (note: This works for INTEGER BASIC, the addresses throughout this article will have to be changed for APPLESOFT BASIC so as to locate the routine at some unused area such as 300 hex = 768 decimal):

```
10 POKE 0,169: POKE 2,162: POKE 4,160: POKE 6,32: POKE 9,133:
   POKE 10,1: POKE 11,134: POKE 12,3: POKE 13,132: POKE 14,5:
   POKE 15,96
```

To make the BASIC interface routine general purpose, four reserved variables will be used: A, X, Y and PC$. Internally, the routine also uses Z and Z$. On entry to the BASIC subroutine that is defined below, the four variables will define the values to be used for the A, X and Y registers; PC$ will contain the ASCII characters representing a 4-digit hexadecimal address of the 6502 routine that is to be called. In the program initialization along with the POKEs that store the machine language program above, PC$ has to be initialized.... 20 DIM PC$(4). The BASIC subroutine that sets up the values for the machine language program looks like:

```
1000 REM BASIC - 6502 INTERFACE CALLED WITH A,X,Y & PC$
1001 Z$ = PC$(4,4): GOSUB 1010: POKE 7,Z:: Z$ = PC$(3,3): GOSUB
     1010: POKE 7,PEEK(7) + Z*16
1002 Z$ = PC$(2,2): GOSUB 1010: POKE 8,Z: Z$ = PC$(1,1): GOSUB
     1010: POKE 8,PEEK(8)+ Z*16
1003 POKE 1,A: POKE 3,X: POKE 5,Y
1004 CALL 0
1005 A = PEEK(1): X = PEEK(3): Y = PEEK(5):RETURN
1010 Z = ASC(Z$) - 176: IF Z > 9 THEN Z = A - 7:RETURN
```

A typical call would be:

```
100 A = value to put in A register: X = x value: Y = y value: PC$ = "four
    hex digits": GOSUB 1001
```

If the same machine language routine is to be called over and over, the JSR address can be left alone unchanged (at a savings in execution time) by calling the routine at GOSUB 1003 after having called it once at 1001 to set up the initial JSR address. Upon return from the subroutine, A, X and Y will contain the values that came back from the machine language that was called. A short BASIC routine can be written to explore and experiment with Monitor routines as simply as the following:

```
150 INPUT "ROUTINE ENTRY POINT (4 HEX DIGITS) ?", PC$
160 INPUT "INITIAL REGISTER CONTENTS (A,X,Y)",A,X,Y
170 GOSUB 1001
180 PRINT "ENTRY "; PC$:" RETURNED A=";A;" X=";X;" Y=";Y:
    GOTO 150
```

This routine can be used to try out many of the routines to see which registers are destroyed during a call and also to verify what the routine does. In some routines more than just the registers needs to be set up, and for those this would have to be expanded. As examples, the following BASIC statements show the usage of some Monitor routines using the calling sequence. As you can see, it is not necessary to set up variables for input registers that are not used for

a particular routine. If you have any questions, catch me at the next Club meeting.

```
200 PC$ = "FF3A": GOSUB 1001: REM BEEP!
210 PC$ = "FC58": GOSUB 1001: REM HOME AND CLEAR = CALL -936
220 A = 10:PC$ = "FB5B": GOSUB 1001: REM VTAB TO LINE 11
230 DIM MSG$(80): MSG$ = "THIS SHOWS CHARACTER OUTPUT"
240 FOR I = 1 TO LEN(MSG$): A = ASC(MSG$(I,I)): PC$ = "FDED":
    GOSUB 1001: NEXT I
etc.
```

Next month I hope to have completely worked out an upgraded version of Don Williams' Integer BASIC to Monitor Floating Point routines. They are covered in the PEEKING AT CALL APPLE, but I hope to make the interface easier and put the square root routine into machine language. Until next month, Pax.

TO GET THE "MOD" FUNCTION WHILE USING APPLESOFT, by Mark Crosby

Use the Function Statement: DEF FN MD(X) = X - INT (X/256) * 256 (MOD 256) Then you can use the function - for example: POKE A, FN MD(X) (you must, of course, set X = to a number).
Or to get X = A MOD B: DEF FN MD (A) = A - INT (X/B) * B (MOD B) then type X=FN MD(A).

An example:

```
10  DEF FN (MD(A) = A - INT (A/256) * 256
20 X = 4096: A=286: REM (FOR RAM, USE X=8192 OR HIGHER)
30 POKE X, FN MD(A)
40 POKE X+1, A/256
50 REM  NOW READ BACK THE POKED NUMBER
100 PRINT PEEK(X)+PEEK(X+1) * 256
110 REM THIS SHOULD PRINT "286"
120 END
```

EXCHANGINGING NEWSLETTERS, by Bernie Urban

The following groups have agreed to exchange newsletters with us. A thanks to them for their cooperation .

Chesapeake Microcomputer Club - M. Alexander
AMRAD - P. Rinaldi
Association of Personal Computer Users - D. Schor
ABACUS (Assn. of Bay Area Computer Users) - Ed Avilar

Copies of their newsletters will be available in the GWU Library.

TRAINING & DEVELOPMENT SESSIONS

At our next meeting, Washington Apple Pi will inaugurate a Training & Development session dealing with topics you want us to cover. We hope you will make a special effort to attend. We hope to cover a different topic each meeting. You are encouraged to participate in a dialogue with the person chairing the session so that you can share your own ideas and hear ideas that others have regarding the topic being discussed. An important part of the session will be feedback from you as to why you like it or not. Please be sure to attend and fill out the forms provided. The first of these sessions (on Saturday, April 28) will be:

"PROGRAMMING IN 6502 MACHINE LANGUAGE" - chaired by John Moon

For those of you who want to learn another language, here is your opportunity to learn the "art" of machine language programming for the APPLE. Machine language programs can run many times faster than BASIC programs. If you have a limited amount of memory available, this will surely be a help.

CALENDAR OF EVENTS

Date	Events/Meetings	For Further Info. Call
April 23	Chesapeake Microcomputer Club 7:30 PM, White Oak Library	Mani Alexander Off. 452-5232
April 26	NOVAPPLE 7:30 PM, St. Stephens United Meth. Church	Jim Nielsen Off. 693-7530
May 9	Assn. of Personal Computer Users 7:30 PM, Chevy Chase Library	Daphne Schor Off. 544-8530
May 17 & 18	Conf. on Microcomputers in Education & Training, Pentagon Quality Inn, Arlington, Va.	Ray Fox Off. (703) 347-0055
May 25	Computerland APPLE Users Group 7:30 PM, Computerland (New location)	Kim Brennan Off. 948-7676

NEXT MEETING OF WASHINGTON APPLE PI
Saturday, April 28, 9:30 AM

GEORGE WASHINGTON UNIVERSITY
Thompkins Hall - School of Engineering, Room 206
23rd & H Streets, NW
Parking roulette, or in students' parking lot, if chains are down.
Convenient to Metro

AGENDA

9:30 - 10:15	Business Meeting - * Constitution and By-Laws * Progress on Incorporation * Solicitation of Ads for the Newsletter * Exchanging Newsletters * Copyright Issue * Discussion of Nomination of Officers * New Business
10:15 - 11:15	Training & Development Session: 6502 Machine Language
11:15 -	Show & Tell; Exchanging of Programs