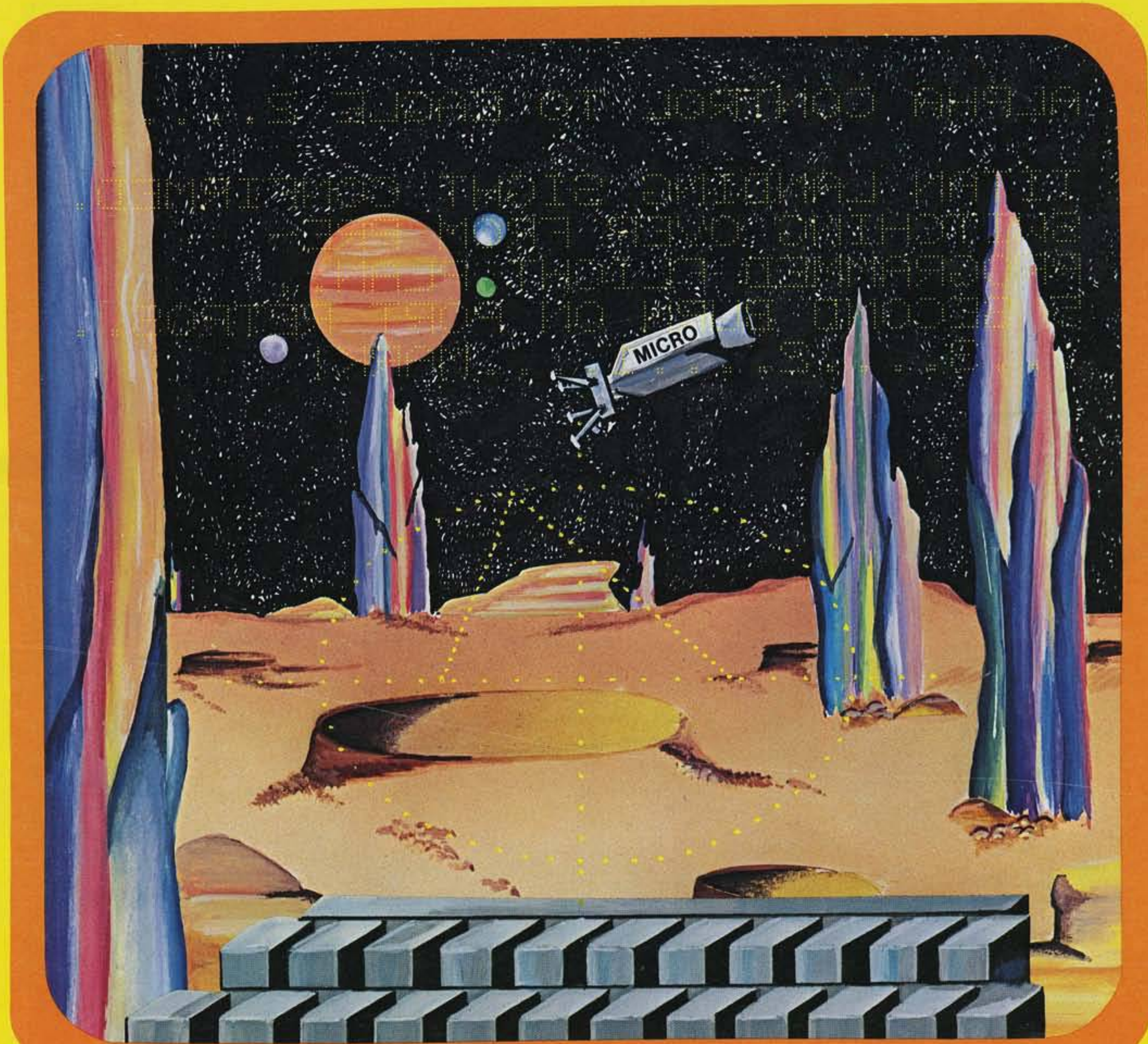
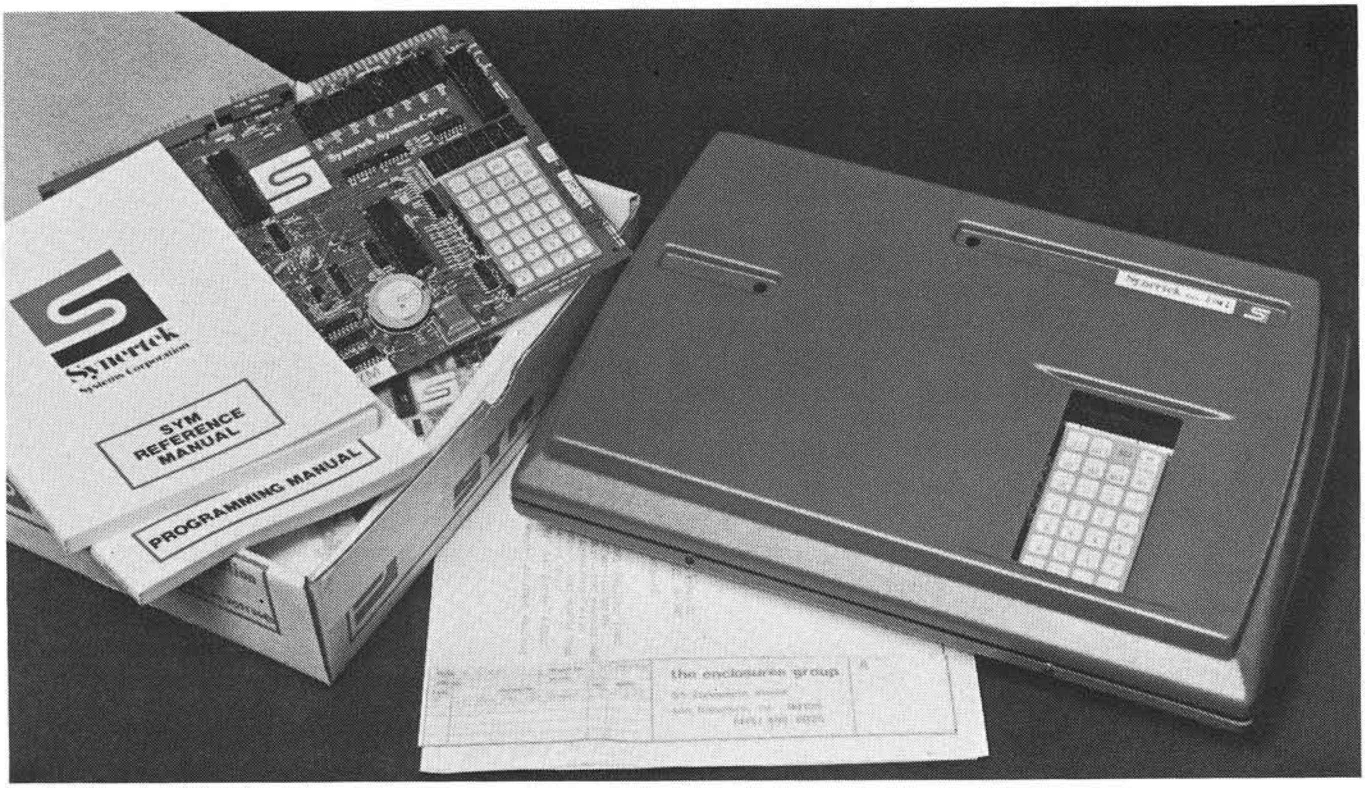
MICRO

THE 6502 JOURNAL



No.27 AUGUST $2.00

SYM-PLICITY



ENGINEERED SPECIFICALLY FOR THE SYM-1 MICRO COMPUTER
- Easy Access for Adjustments
- Room for Expansion
- Protects Vital Components

EASILY ASSEMBLED
- Absolutely No Alteration of SYM-1 Required
- All Fasteners Provided
- Goes Together in Minutes

MADE OF HIGH IMPACT STRENGTH THERMOFORMED PLASTIC
- Kydex 100*
- Durable
- Molded-In Color
- Non-Conductive

AVAILABLE FROM STOCK
- Allow Two to Three Weeks for Processing and Delivery
- No COD's Please
- Dealer Inquiries Invited

ATTRACTIVE FUNCTIONAL PACKAGE
- Professional Appearance
- Popular "Data Blue" Color
- Improves Man/Machine Interface

TO ORDER: 1. Fill in this Coupon (Print or Type Please)
2. Attach Check or Money Order and Mail to:

NAME ____________________________

STREET __________________________

CITY ____________________________

STATE ______________ ZIP ______________

Please Ship Prepaid _____ SSE 1-1(s) @ $36.75 each
California Residents Please Pay $39.14 (Includes Sales Tax)

enclosures group

771 bush street
san francisco, california 94108

* TM Rohm & Haas

Patent Applied For

MICRO™

August 1980
Issue Number 27

Staff

MICRO™ is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824
Publication Number: COTR 395770

Subscription rates: U.S.	$15.00 per year
Foreign surface mail	$18.00 per year
Central America air	$27.00 per year
Europe/Japan air	$39.00 per year
Other air mail	$33.00 per year

For back issues, subscriptions, change of address or other information, write to:
MICRO, P.O. Box 6502,
Chelmsford, MA 01824
or call
617/256-5515

Uses and Abuses of the 6502

When **MICRO** started publication in 1977, the 6502 world was very different from what it is today. At that time you could choose from an OSI board, a KIM-1, or an offering from some small company, many of which are no longer around. Or, you could get in line for one of those brand new systems: the Apple or the PET, which were just being released in limited quantities. Or, you could build a system from scratch. Then, once you had your 6502 based system, you could start hunting for support! It was not easy to find support for the 6502 back in 1977. You might have found an article relating to the 6502 in every other issue of Byte or Kilobaud. There were very few software packages or peripherals available. In every way, the 6502 enthusiast in 1977 was a true pioneer.

Since then, of course, almost everything has changed. The introduction of the Apple and PET brought the 6502 into the spotlight and opened the 6502 world to a new wave of settlers. Instead of a few thousand 6502-based systems in existance, the numbers rapidly grew to the hundred thousands. The major computer magazines started serious coverage of the 6502, and a number of specialized magazines and newsletters covering just the Apple, PET, or other single system emerged. The amount of secondary support in the form of software, books, peripherals, and the like, expanded very rapidly.

While this pattern of growth has been generally positive, there have been some drawbacks. The early 6502 owners were generally knowledgeable about computers and/or electronics and were capable of determining the worth of the various limited offerings. Many of the new users that have been attracted by the Apple and PET computers are relative novices, and with the vast numbers of competing products being offered, many are not in a position to judge the merits of the products. Unfortunately, not all of the products available are worthwhile. With the growth of the 6502 market, elements have been introduced that are much more interested in making the 'fast buck' than in supplying a quality product at a reasonable price. This is probably no more true in the 6502 market than in the microcomputer market in general.

The problem would go away completely if there were some way to have accurate, unbiased, complete evaluations about each product. The purchaser obviously can not rely solely on advertisements, product announcements, or product literature produced by the manufacturer. Independent reviews are probably the best method for getting accurate information out, but a truly **independent** review is very difficult to obtain. Many highly qualified authors who write about the 6502 are so involved in the 6502 world that they have built-in biases, some obvious (as when the individual has his own company) and some less obvious (where a special relationship may have naturally evolved between an author and a producer). Unsolicited reviews tend to be biased since they usually stem from one of two reasons: the author loves a product or the author hates a product. It is difficult to get a qualified evaluator together with a product that should be evaluated. The solution which **MICRO** has implemented is the new feature which appears for the first time in this issue, the **MICROScope**.

In November 1979 we requested that readers sign-up to do reviews for **MICRO**. Several hundred readers responded and we now have a good pool of reviewers to choose from. In December 1979 we printed a form on which manufacturers could request that their products be reviewed. Since then we have generated the necessary paperwork, contacted various reviewers and manufacturers, and have gotten some reviews underway. We will present the results of the reviews in a standard format to make the information easy to use. We have taken every step that we could think of to insure the accuracy and value of the review. Please read the first review which appears on page 31 of this issue, and also read the **Review's Responsibility** information which appears on page 78. We welcome your response to this project. Is this type of review is worthwhile to you? What particular products would you like to see reviewed? Would you like to be a reviewer? Do you have a product of your own that you would like reviewed? Please let us hear from you on this important project. Send correspondence to:

MICROScope, P.O. Box 6502, Chelmsford, MA 01824

Robert M. Tripp

Out of this World

*While some people call this the **Atomic Age**, and others refer to it as the **Space Age**, we all know that it is actually the **Computer Age**. No one will deny the importance of nuclear energy and space exploration and technology. Their impact on our lives is currently, however, very insignificant in relation to the computer.*

*There is, undeniably, a common interest to computers and space. They have both been used as cornerstones for science fiction, and there are numerous space oriented computer games: **Star Trek, Space Ace, Lunar Lander,** dating back at least to the space war game on the PDP-1.*

The cover depicts the computer on a distant planet. While it will probably be some time before an Apple, PET, or other 6502 based microcomputer system lands in such a distant world, there are some interesting space related uses for the microcomputer systems today. Two articles in this month's issue deal with space. One provides a program for generating the set of parameters required for tracking satellites; the other generates a map of the solar system for specified planets over specified periods of time.

μ

Data Statements Revisited

The power of BASIC can be greatly enhanced by the ability of a program to update program statements. This article discusses the fundamentals of the technique and presents detailed program examples.

Virginia Lee Brady
6 Reldas Ct. Apt. L
Cockeysville, MD 21030

Since I began working with data statements, I have found that just poking them into memory is frequently not enough. In many applications it is necessary to update the actual statements in memory so that they can be used in subsequent runs of the program. This became especially evident when I was working on a directory program using linked lists. I had worked out a program to handle insertions and deletions for string arrays (i.e.-changing the "links"), but these revisions were only good for the duration of the run of the program; afterwards, the data statements were still in their original form. In order for the program to be truly useful, I soon found that I had to change the physical lines in memory so that these new lines could be saved along with the program, and when the program was rerun, the arrays would contain the corrected strings and links. While I specifically developed these routines for use in a linked list application, the procedures involved in changing program lines that can be used in other ways.

Although this is not a tutorial on linked lists per se, I should mention some of the concepts involved to show how they can be implemented in Applesoft. The program basically uses each individual data line as an atom or record within a specific "file" (the directory). The fields are those elements separated by commas. I set up my data statements to look similar to what is found in Figure 1.

Thus the directory is a two-dimensional array whose fields include last name, first name, and address. There is also a seperate numeric array, NXT%(I), that is used to hold the links to the next greater entry. If NXT%(I) equals zero, the RECORD(I),is the greatest one on the list, although not necessarily the last one. There is also a line defining two more variables:

```
20 X% =003: HEAD% =1
```

HEAD% is the first or lowest record in the list; alphabetically, Aardvark comes before any of the other names. X% is the total number of records in the file. A new entry would be placed at position X% +1. Since this is the update position, it is referred to as UP.

It should be noted, then, that for any RECORD(I), there are two other records that are in some way "previous" to it:

1) PREV(ious) is a value which is set up by the linked list algorithm and indicates the next "smaller" record.
2) RECORD(I-1) is the record immediately before RECORD(I), both in the string array and sequentially in memory.

The alphabetical integrity of the list is maintained by the links. This is similar to "follow-the-dots" in which one goes from 1 to 2 to 3..., wherever the dots occur. When a new record is entered, it is placed at the end of the list at the update position, and the two links are changed. The next smallest record's link field, NXT%(PREV), is set to point to the new entry, RECORD (UP), and the new record's link points to its successor, which is the entry that NXT%(PREV) used to point to. In the program example, when Collins is added, PREV is assigned a value of two because RECORD(2), Brady, is the next smallest record. Then, in line 525, NXT%(4) is assigned the value of NXT%(2), which points to Zebra, and NXT%(2) is set to the update position. This way, when the links are followed, the chain goes from 1 to 2 to 4 to 3, so that the list is kept in alphabetical order. Similarly, if a deletion occurs for Brady, RECORD(2), PREV is assigned a value of one, the next smallest record. In line 630 NXT%(1) is set to equal NXT%(2) so that the chain goes form 1 to 4 to 3, completely skipping past Brady. (If you've never used linked lists before, try following this by hand to get an idea of how it works.)

The actual steps involved in calculating the values of NXT%(I), HEAD%, AND PREV are beyond the scope of this article. My purpose is to explain how to get the information into the Applesoft program once a suitable algorithm has been developed by the user. It is assumed that the user is familiar with the idea of poking data lines into a program so that these updates can be performed under program control, rather than by hand on the user's part. (See Micro 19:44)

(I)	Line	DIR$(I,1)	DIR$(I,2)	DIR$(I,3)	NXT%(I)
(1)	10001 DATA	AARDVARK,	SAM,	ANYWHERE,	002
(2)	10002 DATA	BRADY,	VIRGINIA,	SALISBURY,	003
(3)	10003 DATA	ZEBRA,	TED,	CITY ZOO,	000

Figure 1

Reading

During the reading in of a data line, several events take place beneath the surface of Applesoft. Locations $7D.7E (125-126) are set to the address of the 00 byte which indicates the end of a particular statement read in. This pointer acts as a sort of "place holder" in that it always points to the last data statement that was read in. When Applesoft next encounters a READ statement, it begins its search from this address and the pointer is updated to the next "end -of- record" mark. These locations can be used to the programmer's advantage if they are saved in a data pointer array, so that after all of the data has been read in, each element in the DP(I) array points to the end-of-record mark for each corresponding RECORD(I). Using these locations it is then possible to calculate backwards in order to reference a specific part of a data line. If NXT%(I) is a three byte link field placed at the end of the line (the most convenient location), then DP(I)-3 is the start of that field and DP(I)-4 is its preceding comma. Conversely, DP(I-1) is the end of the line immediately preceding RECORD(I), and from there you can calculate the beginning of line I.

Another pair of pointers set up during the read are $7B.7C (123-124). These contain the line numbers of the last data line which was read in. Calculating Line = PEEK(123) + PEEK(124)*256 immediately after the main read loops will mean that LINE is always equal to the highest numbered data line. Everytime a new line is poked into place, this value needs to be incremented so that each data line has a unique line number.

Writing

To understand how a line that is physically stored in RAM can be manipulated, it is necessary to review the anatomy of the line from the interpretter's point of view:

(line 1) 10003 DATA ZEBRA, TED, CITY ZOO, 000
(pp pp = A; ll ll = B; 83 = C; (ascii's 2C 30 30 30 = D: 00 = D.)
DP(I-1)pointer line "data" DP(1-4) DP(I)

```
]RUN
THE DIRECTORY :
AARDVARK SAM
ANYWHERE
BRADY VIRGINIA
SALISBURY
ZEBRA TED
CITY ZOO
NEXT WE WILL SHOW THE LINES THAT HAVE
 BEEN CHANGED. ANY KEY TO CONTINUE
20 X% = 003:HEAD% = 1
10001  DATA AARDVARK,SAM,ANYWHERE
     ,002
10002  DATA BRADY,VIRGINIA,SALISB
     URY,003
10003  DATA ZEBRA,TED,CITY ZOO,00
     0
ANY KEY TO CONTINUE
               INSERT
WE WILL NOW INSERT :
COLLINS,BILL,SALISBURY
THE DIRECTORY :
AARDVARK SAM
ANYWHERE
BRADY VIRGINIA
SALISBURY
COLLINS BILL
SALISBURY
ZEBRA TED
CITY ZOO
NEXT WE WILL SHOW THE LINES THAT HAVE
 BEEN CHANGED. ANY KEY TO CONTINUE
20 X% = 004:HEAD% = 1
10001  DATA AARDVARK,SAM,ANYWHERE
     ,002
10002  DATA BRADY,VIRGINIA,SALISB
     URY,004
10003  DATA ZEBRA,TED,CITY ZOO,00
     0
10004  DATA COLLINS,BILL,SALISBUR
     Y                  ,003
ANY KEY TO CONTINUE
               INSERT
WE WILL NOW INSERT :
MICRO,MAGAZINE,CHELMSFORD
THE DIRECTORY :
AARDVARK SAM
ANYWHERE
BRADY VIRGINIA
SALISBURY
COLLINS BILL
SALISBURY
MICRO MAGAZINE
CHELMSFORD
ZEBRA TED
CITY ZOO
```

A. 2 bytes — pointer to next line of Basic (to next pointer)
B. 2 bytes — hex equivalent of the line number
C. 1 byte — "83" — token for "DATA"
D. N bytes — ASCII equivalents of the program line
E. 1 byte — "00" — indicates the end of the line

All of the information stored within the data statement is stored as ASCII codes, since at the time it is entered, Applesoft does not know whether it will be read into a string or numeric variable. Therefore even NXT%(I) is stored as an ASCII equivalent rather than a hexadecimal equivalent of the value. Because of this, the value can be manipulated as a string [NXT$ = STR$(NXT%(I))] and poked into place to update the value of any NXT%(I).

In my set up, this link field is exactly three characters long, allowing up to 1000 combinations of numbers. The leading zeros are used to allow room for expanding to a larger number, say, from nine to ten. There are two ways to set up NXT% with the leading zeros:

1) It can be done as a loop:

100 IF LEN(NXT$) < 3 THEN NXT$ = "0" + NXY$: GOTO 100

This limits NXT$ to a length of exactly three, but has a drawback in time. Whenever Applesoft encounters a GOTO instruction, it starts at the smallest line number and executes a sequential search for the specified line number. If the line is 100 the Applesoft may have to search through up to 100 lines before it finds the right one. If the above instruction causes Applesoft to loop twice, this is 200 lines it may have to search through and this takes time.

2) A quicker way is with a sequence of instructions like:

100 NXT$ = "00" + NXT$ (now length is 3,4,or 5)
110 NXT$ = RIGHT$(NXT$,3) (now equals its own righthand side)

Getting NXT$ into place for RECORD(UP) is easy; it is concatenated to the record string and poked into place at the same time as the rest of the record. Changing the NXT% (PREV) is a bit more complex; it needs to replace the old NXT%(PREV) in memory as precisely as a piece of stone is replaced in a mosaic. Assuming that you have already set up a DP array, the this string fits into locations DP(PREV)-3, DP(PREV)-2, and DP(PREV)-1. This can be done in a loop, where DP = DP(PREV)-4, the

```
NEXT WE WILL SHOW THE LINES THAT HAVE
 BEEN CHANGED. ANY KEY TO CONTINUE
20 X% = 005:HEAD% = 1
10001  DATA AARDVARK,SAM,ANYWHERE
     ,002
10002  DATA BRADY,VIRGINIA,SALISB
     URY,004
10003  DATA ZEBRA,TED,CITY ZOO,00
     0
10004  DATA COLLINS,BILL,SALISBUR
     Y                    ,005
10005  DATA MICRO,MAGAZINE,CHELMS
     FORD                 ,003
ANY KEY TO CONTINUE
NOW DELETING :
BRADY
VIRGINIA
SALISBURY
THE DIRECTORY :
AARDVARK SAM
ANYWHERE
COLLINS BILL
SALISBURY
MICRO MAGAZINE
CHELMSFORD
ZEBRA TED
CITY ZOO
NEXT WE WILL SHOW THE LINES THAT HAVE
 BEEN CHANGED. ANY KEY TO CONTINUE
20 X% = 005:HEAD% = 1
10001  DATA AARDVARK,SAM,ANYWHERE
     ,004
10002  DATA BRADY,VIRGINIA,SALISB
     URY,004
10003  DATA ZEBRA,TED,CITY ZOO,00
     0
10004  DATA COLLINS,BILL,SALISBUR
     Y                    ,005
10005  DATA MICRO,MAGAZINE,CHELMS
     FORD                 ,003
ANY KEY TO CONTINUE
]LIST
10  LOMEM: 9999
20 X% = 003:HEAD% = 1

22  REM
     DO NOT CHANGE ANYTHING ABOVE
     THIS WITHOUT RECALCULATING

     LINE 72 !
25  DIM DIR$(100,4),DP(100),NXT%(
     100)
26  GOTO 1000
30  RESTORE : FOR I = 1 TO X%: FOR
     J = 1 TO 3: READ DIR$(I,J): NEXT
     : READ NXT%(I):DP(I) =  PEEK
     (125) +  PEEK (126) * 256: NEXT

35 LINE =  PEEK (123) +  PEEK (12
     4) * 256:UP = I: RETURN
40  REM CHANGE NXT%(PREV)
42 NXT$ = "00" +  STR$ (NXT%(PREV
     )):NXT$ =  RIGHT$ (NXT$,3)
44 DP = DP(PREV) - 4: FOR I = 1 TO
     3: POKE DP + I, ASC ( MID$ (
```

position of the comma, and then poking then new information into place. By using a simple NXT$ variable and changing the value as needed, the same subroutine can be used for a RECORD(I). This is shown in lines 42-44.

Using a variable such as X% as the limit for a FOR/NEXT loop is simpler and faster than using trailers. It eliminates the need for statements such as "IF DIR$(I,1) = ""THEN...", "I = I + 1", and "GOTO...". It also does away with the problems associated with trailers-writing over the old trailers and setting up a new set of trailers every time a record is added. Because the GOTO is avoided, the need for repeated searches for a line is also eliminated. Since X% is a variable, its value is easy to set with a statement of "X% = X%*1", but the value is lost when the variable table is cleared by a RUN. The only way to retain this value is to make the line defining X% a permanent part of the program and then updating it so that when the program is saved, the revised line is also saved and can be interpretted again when the program is run the next time.

The memory locations containing the "003" in line 20 bear a specific (offset) relationship to the beginning of the program. In ROM Applesoft, the starting position is $801 or 2049. In either RAM or ROM Applesoft, the starting position is held in $67.68. If a variable is defined as START = PEEK(103) + PEEK (104)*256, then START plus the predefined offset value is the location of the first zero. Determining this offset value is fairly straight foreward — go into the monitor and look for the "003." Then count how far it is from the beginning of the program to this location. What you are looking for is:

(line) 20 X % = 0 0 3
(monitor) pp pp 14 00 58 25 DO 30 30 33...

This process can be simplified by placing this line as near the beginning of the program as possible after adjusting LOMEM, and while there are few lines following it. If you are reasonably sure of the approximate offset value, you can also try this in an immediate mode:

```
     NXT$,I)): NEXT : RETURN
60  REM CALC HI/LO BYTES
65 HI =  INT (NO / 256):LO = (NO /
     256 - HI) * 256: RETURN
70  REM CHANGING X%
72 X$ = "00" +  STR$ (X%):X$ =  RIGHT$
     (X$,3): FOR I = 1 TO 3: POKE
     START + 16 + I, ASC ( MID$ (
     X$,I)): NEXT : RETURN
80  LIST 20: LIST 10001 - 10010: PRINT
     "ANY KEY TO CONTINUE": HTAB
     20: GET R$: RETURN
400  REM  PRINT ROUTINE
410 INDEX = HEAD%
420  HOME : PRINT "THE DIRECTORY
     :": PRINT : PRINT
430  PRINT DIR$(INDEX,1); SPC( 1)
     DIR$(INDEX,2): PRINT DIR$(IN
     DEX,3)
435  PRINT
440 INDEX = NXT%(INDEX): IF INDEX
      <  > 0 THEN 430
450  PRINT "NEXT WE WILL SHOW THE
      LINES THAT HAVE   BEEN CHAN
     GED. ANY KEY TO CONTINUE": HTAB
     20: GET R$: RETURN

500  REM INSERT
502  REM IN A USER PROGRAM THE
     VALUE OF PREV WOULD BE
     SUPPLIED BY THE ALGORITHM
503  REM  ALSO BEWARE OF INSERTS
     THAT CHANGE THE VALUE OF THE
     VARIABLE 'HEAD%'
504  REM OR OF AN INSERT THAT
     WOULD BE THE LAST ONE ON THE
     LIST ('000')
510  HOME : PRINT  TAB( 15)"INSER
     T": PRINT : PRINT
520  PRINT "WE WILL NOW INSERT :"
     : PRINT NAME$
525 NXT%(UP) = NXT%(PREV):NXT%(PR
     EV) = UP
530  GOSUB 40
535 NXT$ = "00" +  STR$ (NXT%(UP)
     ):NXT$ =  RIGHT$ (NXT$,3)
540  IF  LEN (NAME$) < 40 THEN NA
     ME$ = NAME$ + " ": GOTO 540
550 NAME$ = NAME$ + "," + NXT$
560 LINE = LINE + 1:NO = LINE: GOSUB
     60
565 PSN =  PEEK (175) +  PEEK (17
     6) * 256: POKE PSN,LO: POKE
     PSN + 1,HI: POKE PSN + 2,131

570  FOR I = 1 TO  LEN (NAME$): POKE
     PSN + I + 2, ASC ( MID$ (NAM
     E$,I)): NEXT : POKE PSN + I +
     2,0
575 NO = PSN + I + 3: GOSUB 60: POKE
     PSN - 2,LO: POKE PSN - 1,HI:
      POKE NO,0: POKE NO + 1,0
577 NO = NO + 2: GOSUB 60: POKE 1
     75,LO: POKE 176,HI
580 X% = X% + 1: GOSUB 70
585  GOSUB 30
590  FOR T = 1 TO 1000: NEXT : RETURN
```

S = 2049
FOR I =0 TO 30; PRINT I, CHR$(PEEK(S + I)): NEXT

This will generate a listing of:

.
.
.
.

```
17    0
18    0
19    3
.
.
.
```

In this example, the offset would be 17, so poking would begin at START + 16. Getting the value into the line is now handled the same as NXT$ was (see line 72).

Some words of caution are in order at this point. Once the offset value of the X% line has been established, adding or deleting a single characterin a preceding line will cause X$ to be poked into the wrong place (and give you a very strange listing.) This can sometimes be remedied by typing "20" to erase the line and then type it over again. Similarly, if NXT$ is poked into the wrong place, it will cause problems such as overwriting its preceding comma or the succeding end-of-line mark. Remember that once the new lines are poked into memory, they are a permanent part of the program and are not cleared by a new run of the program. This means that if you manually delete a data line, you must also manually change the X%. If LINE is set to a constant, rather than to the last value in locations $7B.7C, then when the program is rerun, the same line numbers will be used over and over. (While this is an interesting effect, it quickly loses its appeal when you try to delete the third ocurrence line 10003.) Be forewarned that poking values into inappropriate places is a fast way to demolish a program and probably Applesoft. For this reason, I would advise you to save a copy of the program after you have typed it in and before you try to run it. That way if it bombs you can re-load it and find the error before running it again.

Note that line 540 is not necessary to this version of the program; it merely assures that all data lines are the same length, and leaves room in case the line is later changed by the program. (An application not included in this program.) In fact, 540 is a relic from an earlier version of the program and could be replaced by a statement to concatenate NAME$ to a long string of spaces and then truncating it to an appropriate length. This is left as an exercise to the reader. Line 80 is also unnecessary; it is included to list the various lines as they change, so that the user does not have to drop out of the program to examine the lines in question.

The program listing incorporates these routines and those from "Data Statement Generator" and will produce an example of a linked list in Applesoft. Although the program has relatively few fields, the arrays could be expanded to accomodate any number of fields depending on the application. Calculating the values of HEAD%, PREV, and NXT%(I) are left to the user, as are the additions of subroutines for sorting, searching and inputting the various fields.

There is a striking similarily between the way data statements are stored in RAM and the way Applesoft text files are written to disk. The routines in this article could serve as a vehicle for learning about text files, and have an added advantage in that their results are easier to examine. Actually this whole directory could have been handled with a text file on a disk, but using this routine and tape is about $500 cheaper.

μ

```
600  REM DELETE ROUTINE
602  REM IN A PROGRAM, THE VALUES
     OF LOC(ATION) & PREV(IOUS)
     WOULD BE SUPPLIED BY THE
     DELETION ALGORITHM.
603  REM DELETING THE ENTRY POINT
     ED TO BY HEAD% WILL REQUIRE

     EXTRA UPDATES.
610  HOME : PRINT "NOW DELETING :
     "

620  FOR I = 1 TO 3: PRINT DIR$(L
     OC,I): NEXT
630 NXT%(PREV) = NXT%(LOC): GOSUB
     40
640  FOR T = 1 TO 1000: NEXT
650  RETURN
1000  REM MAIN LOOP
1002 START =  PEEK (103) +  PEEK
     (104) * 256
1005  GOSUB 30
1010  GOSUB 400: LIST 20: LIST 10
     001 - 10003: PRINT "ANY KEY
     TO CONTINUE": HTAB 20: GET R
     $
1020 NAME$ = "COLLINS,BILL,SALISB
     URY":PREV = 2
1030  GOSUB 500
1040  GOSUB 400: GOSUB 80
1050 NAME$ = "MICRO,MAGAZINE,CHEL
     MSFORD":PREV = 4
1060  GOSUB 500: GOSUB 400: GOSUB
     80
1070 NAME$ = "BRADY":LOC = 2:PREV
      = 1
1080  GOSUB 600
1090  GOSUB 400: GOSUB 80



2000  REM VIRGINIA LEE BRADY
2002  REM ROUTINE TO SHOW HOW TO
      CHANGE APPLESOFT LINES
      UNDER PROGRAM CONTROL.
10001  DATA AARDVARK,SAM,ANYWHERE
     ,002
10002  DATA BRADY,VIRGINIA,SALISB
     URY,003
10003  DATA ZEBRA,TED,CITY ZOO,00
     0
```

Satellite Tracking with the AIM — 65

Here is a useful application program for the astonomy buff. It calculates the information required for tracking any satellite. Written for the AIM 65, it may be easily adapted to any other 6502 with BASIC.

C.R. MacCluer
P.O. Box 1858
E. Lansing, MI. 48823

The program listed below will supply the user with all necessary tracking data for any earth satellite. The program as printed below is set up to track the AMSAT-Oscar Phase III-A which was due to be launched May, 1980 by the European Space Agency with their new Ariane from French Guiana.

To modify this program for other satellites merely change the parameters defined in line 110:

A = length of the semimajor axis (in km)
E = eccentricity
P = period of revolution (in minutes)
K = inclination of orbital plane (in degrees)
W = argument of perigee (in degrees).

For an explanation of these terms see my forthcomming HAM RADIO article "The Geometry of Phase III-A".

Elliptical orbits will precess due to the oblateness of the earth, that is, the argument W of peri ee will change over time. For instance it is expected that Phase III-A will precess .07 degrees per day and thus W will have to be up-dated monthly.

For circular orbits take A = 6371 +(height of satellite in km) and E = W = 0.

This program as it stands is set up to printout in 10 minute increments, which is perfectly fine enough for satellites such as Phase III-A with periods of eleven hours or more. But for near earth orbits such as AMSAT-Oscar 8, increments of 1 minute are preferred. To obtain minute by minute printouts, merely delete the "STEP 10" in line 300. So for example to track AMSAT-Oscar 8 make the following changes:

```
100 A=7281:E=0:P=103:K=99:W=300:FOR
T=0 TO P
```

The subroutines 4000 and 5000 are Arccosine and Arctangent routines respectively.

One last change. In line 150 are entered the latitude L3 and longitude L4 (in degrees) of E. Lansing, Michigan. You will of course change these values to match your location.

```
        ROCKWELL     AIM     65

 100 PI=3.141592654
 110 A=24313: E=67627: P=628.83:
K=57: W=210
 120 A=A/6371: N=2*PI/P:
E9=SQR((1+E)/(1-E))
 130 DEF FNF(X)=X*PI/180
 140 DEF FNG(X)=X*180/PI
 150 L3=42.75: L4=84.5
 160 K=FNF(K): W=FNF(W):
L3=FNF(L3): L4=FNF(L4)
 170 K1=COS(K): K2=SIN(K):
G1=COS(L3): G2=SIN(L3)
 180 PRINT ''TIME OF EQX IN UTC?
   (FORMAT 0000)''
 190 INPUT TO
 200 T9=TO/60-2*INT(TO/100)/3
 210 PRINT ''LONGITUDE OF EQX?
          (DECIMAL  DEGREES)''
      220        INPUT        00
 225 PRINT
 230 00=FNF(00): VO=2*PI-W:
EO=TAN(VO/2)/E9
 240 X=EO
 245 PRINT
 250 GOSUB 5000
 260 EO=2*Y
 270 IF VO<PI THEN 290
 280 EO=2*PI+EO
 290 MO=EO-E*SIN(EO): MI=MO:
E1=EO: P1=A*(1-E*E)
 295 PRINT ''UTC  AZ EL LAT LNG''
 300 FOR T=-INT(MO/N)TOP STEP 10
 310 01=00-L4+T*PI/720
 320 Z1=G1*COS(01): Z2=G1*SIN(01):
Z3=G2
 330 M=T*N+MO
 340 D=(M-M1)/:01-E*COS(E1)):
E1=E1+D: M1=E1-E*SIN(E1)
 350 IF ABS(M-M1)>102-8 THEN 340
 360 X=E9*TAN(E1/2)
 370 GOSUB 5000
 380 V=2*Y
 390 IF E1<PI THEN 410
 400 V=2*PI+V
 410 R=PI/(1+E*COS(V))
 420 S1=R*COS(V-VO):
S2=K1*R*SIN(V-VO):
S3=K2*R*SIN(V-VO)
 430 D=S1*Z1+S2*Z2+S3*Z3:
X=(D-1)/SQR(R*R-2*D+1)
 440 GOSUB 4000
 450 B1=PI/2-Y
 460 IF B1=>0 THEN 470 NEXT T
 475 END
 480 B1=INT(FNG(B1))
 490 X=(S3-Z3*D)/(SQR(1-Z3*Z3)
*SQR(R*R-D*D))
 500 GOSUB 4000
 510 A1=Y: H=S2*Z1-S1*Z2
 520 IF H>0 THEN 540
 530 A1=2*PI-A1
 540 A1=INT(FNG(A1))
 550 X=SQR(1-(S3/R)*(S3/R))
 560 GOSUB 4000
 570 N1=INT(FNG(SGN(S3)*Y))
 580 H=S2*COS(01+L4)-S1
*SIN(01+L4)
 590 U=S1*COS(01+L4)+S2
*SIN(01+L4)
 600 X=U/SQR(S1*S1+S2*S2)
```

To use this routine, load and run, answer the two questions posed (the time and longitude of the ascending equator crossing) and the AIM-65 will print out time, azimuth, elevation, as well as the latitude of the subsatellite point in 10 minute increments. I have included a sample printout for AMSAT-Oscar III-A.

Another version of this program has a software clock and a real-time routine which pokes down to dataport B the azimuth and elevation, thus controlling antenna rotors via an interface of my design. A description of the interface will soon appear in 'Ham Radio'. You may obtain this second version from me for the consideration of $5.00 ($4.00 of which is donated to the AMSAT Phase III program), a blank cassette, and a sufficiently stamped, self-addressed carton in which to return the cassette. Send to: C.R. MacCluer, P.O. Box 1858, East Lansing, MI 48823.

```
 610 GOSUB 4000
 620 IF H<0 THEN 640
 630 Y=2*PI-Y
 640 L1=INT(FNG(Y))
 650 T1=T9+T/60: T1=60*T1+40*
INT(T1): T1=INT(T1)
 660 T1*10000+T1:
T1$=RIGHT$(STR$(T1),4)
 670 A1$='' ''+STR$(A1):
A1$=RIGHT$(A1$,4)
 680 B1$='' ''+STR$(B1):
B1$=RIGHT$(B1$,4)
 690 N1$='' ''+STR$(N1):
N1$=RIGHT$(N1$,4)
 700 L1$='' ''+STR$(L1):
L1$=RIGHT$(L1$,4)
 710 PRINT T1$+A1$+B1$+N1$+L1$
 720 NEXT T
 730 END
 4000 Y=1: X1=COS(1)
 4010 DY=(X1-X)/SIN(Y):
Y=Y+DY: X1=COS(Y)
 4020 IF ABS(X1-X)>108-8 THEN 4010
 4030 RETURN
 5000 X2=X: X1/SQR(1+X*X)
 5010 GOSUB 4000
 5020 Y=SGN(X2)*Y: RETURN
```

```
RUN
TIME OF EQX IN UTC?
  (FORMAT 0000)
? 0500
LONGITUDE OF EQX?
(DECIMAL DEGREES)
? 90
```

UTC	AZ	EL	LAT	LNG
0326	183	0	-28	88
0346	182	7	-23	86
0356	181	13	-18	86
0406	181	18	-15	85
0416	182	23	-11	86
0426	182	27	-9	86
0436	184	30	-6	87
0446	185	33	-4	88
0456	187	36	-1	89
0506	189	39	1	90
0516	191	41	3	91
0526	194	43	4	93
0536	196	45	6	94
0546	199	47	8	96
0556	202	48	9	97
0606	206	49	11	99
0616	209	50	12	100
0626	213	51	14	102
0636	216	52	15	103
0646	220	53	16	105
0656	224	53	18	106
0706	228	53	19	108
0716	231	54	20	109
0726	235	54	21	111
0736	239	54	23	112
0746	242	54	24	114
0756	246	53	25	115
0806	249	53	26	117
0816	253	53	27	118
0826	256	52	29	120
0836	259	52	30	121
0846	262	51	31	123
0856	265	51	32	124
0906	268	51	33	125
0916	271	50	35	126
0926	274	50	36	127
0936	276	49	37	128
0946	279	49	39	129
0956	282	49	40	130
1008	284	49	41	131
1016	287	49	43	131
1026	289	49	44	131
1036	292	49	46	131
1046	294	49	47	131
1056	297	50	49	130
1106	300	50	50	129
1116	303	51	52	127
1126	306	53	53	124
1136	310	55	55	120
1146	315	57	56	114
1156	322	60	56	107
1206	332	65	56	98
1216	354	70	54	86
1226	41	70	50	73
1236	84	52	41	59
1246	104	16	26	45

We had planned to use the original AIM listings with this article, but unfortunately, they were blue, and so light that our printer's camera could not pick them up. Therefore, we have typeset the listings ourselves. Hopefully, this will not cause problems. We do caution the user about some of the labels. Because the listings were so light, they were difficult to read while setting the type. So, watch out for O0. It is not double zero, nor double 'oh' it is meant to be 'oh zero'; and P1 or PI (P one and P 'eye').

μ

MICRO Limerick Contest

Much to our delight here at Micro, we have had good response to our limerick contest first announced in the May issue. We had no idea that so many of our readers were poetic!

Almost every entry that we have received to date has been in the required form. A few had to be disqualified because they were too long to be limericks. We must congratulate all who submitted limericks for taking the time and interest in our first contest of this kind. The entries have provided us with something to look forward to with each day's mail, and have been entertaining reading material for the company bulletin board.

We had originally planned to announce the winner of the contest in the September issue, but we are changing that now. Instead, we are publishing what we have judged to be the best entries, and we will let you readers decide.

All we ask is that you write or phone and tell which limerick you like best. The winning limerick, and the author of it will be announced in the October issue.

And now, here are what we have judged to be the best limericks. Somewhere in their midst is the winning one.

There was a young hacker named Drew
Who programmed all day and night too.
By morning 'twas done
But he didn't type run.
The poor little guy entered new.

Art Carpet
Canyon Country, CA

When first introduced to the 6-5-0-2,
I conceded it's clever; but what can it do?
Ask Atari or Apple, they know what you get,
Ask Sym or Kim to find out my Pet.
After taking Aim, I found it true, Micro does it all for you.

Ralph R. Orton
Granada Hills, CA

There once was a key-pounder named Rick
Who thought he knew every trick
That was under the sun
To make programs run
Well Micro improved his perfection, but quick!

W.G. Fullerton
Ottawa, Canada

For the 6516 by Synertek
To Santa Clara, we did trek
But past the editor did slide
An April Fool's by Mr. Hyde
Did our expansion plans thus wreck.

Earl Morris
Midland, MI

A con man of articulate diction
Flattered her beyond description
She responded with laughter
"It's not me that you're after,
But my Micro journal subscription!"

Harold I. Mathis
Southfield, MI

Some magazines are geared too low,
Others are just so-so.
For a 6-5-0-2
About the best you can do
Is subscribe to the best, namely Micro.

Mike Sullivan
Belleville, NJ

In the world of zero and one,
There's a mag which is second to none.
It's Micro, you see
The journal for me,
Or my 6502 would be done.

D. Duckworth
Las Vegas, NV

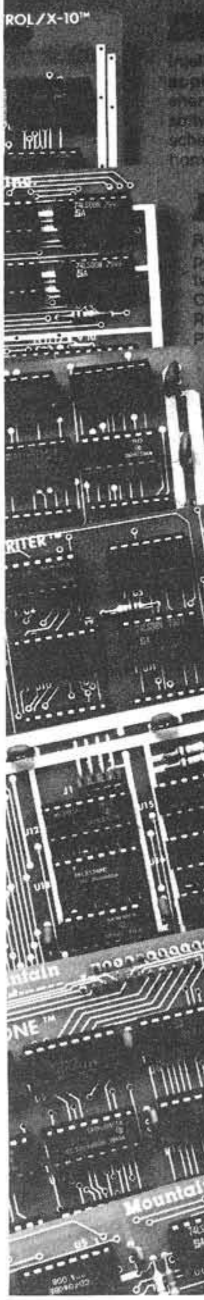
He saw her reading Micro and knew
That she owned a six five o'two.
He sent her a scribble
She gave him a nibble
Now they both share the same CPU.

H. I. Mathis
Southfield, MI

Continued on page 70...

Mountain Hardware makes more peripherals for the Apple Computer than Anybody.

and . . .
a place to put them



NTROL X-10

Intelligent Home Controller for lights and appliances. Real-time schedules and energy conservation. Complete applications software package. Home security with random scheduler. Power usage accounting package for home energy cost control. No wiring required.

APPLE CLOCK

Real-time and date information. Interrupts permit Foreground/Background operation of two programs simultaneously. Battery back-up. Crystal-controlled for ± .001% accuracy. Onboard ROM for easy access from BASICs. Supports PASCAL. Time from one millisecond to one year.

SUPERTALKER SD200

nput/Output Speech Digitizer. Permits talking programs. I/O capability llows interactive programs with speech-prompted inputs. Use output for peech directed activities in business systems, announcements in a control-oom, or sound effects in entertainment programs. Easy to use because input s well as output is under user control with special software operating system.

ROMWRITER

Program your own EPROMs. Create your own firmware. Programs 2K, 2716 5V EPROMs. Disk software package provides easy EPROM programming. EPROMs are verified after BURN. RUN your programs from on-board socket or install them on ROMPLUS+.

ROMPLUS+

More power for your system through firmware. Six sockets accept 2716 EPROMs or ROM equivalents. Six or any combination can be used at once. Scratch-pad RAM and two TTL connectors. Special 2K ROMs available for powerful system enhancement: Keyboard Filter ROM—COPYROM—Others coming soon.

MusicSystem

Sophistication previously available only on experimental mini and mainframe computer synthesizers. Digital instrumental music synthesizer system. 16 voices in stereo. Instrument definitions simulate the sound of real instruments—and more. Fully programmable waveforms. Envelope Control. Composition system—sheet music input using standard music notation. Chords and multi-part scoring up to 16 voices. A true instrument that anyone with an Apple can play.

A/D+D/A

16 channels analog to digital input. 16 channels digital to analog output. Eight bit resolution. Super-fast 8μ sec. conversion time. Monitor and output to the real world. All on one card.

EXPANSION CHASSIS

By popular demand! Eight more slots for your Apple. Attractive sturdy enclosure. Its own heavy duty power supply. Easy to use. Address cards in Expansion Chassis the same way as in your Apple. Only one additional command to specify in Apple or in Expansion Chassis. Compatible with all Apple peripherals.

MOUNTAIN HARDWARE has the most comprehensive line of Apple peripherals available. Anywere. From anybody. We know the Apple inside and out and are committed to providing the most innovative and unique products to expand and enhance its capabilities and use. After all, we were the first company to make an Apple peripheral—except Apple Computer.

The message is simple. If you have an Apple, you need to know MOUNTAIN HARDWARE.

Available at Apple Dealers worldwide.

Mountain Hardware
Leadership in Computer Peripherals
A Division of Mountain Computer, Inc.
300 Harvey West Blvd.
Santa Cruz, CA 95060 (408) 429-8600

MORE PERIPHERALS? Send me information.
NAME ____________
ADDRESS ____________
CITY ________ STATE ________ ZIP ______

Apple is a trademark of Apple Computer Inc.

Better Utilization of APPLE Computer Renumber and Merge Program

The Renumber and Merge program is very useful, and here is a technique that makes it even easier to use.

Frank D. Chipchase
21 St. George St.
West Milford, NJ. 07480

I consider a utility program excellent when it can be utilized at any time under any condition. This brings me to that marvelous Applesoft Renumber and Merge program which comes with DOS 3.2.

Many times, during programming or editing, the need arises to move chunks of your program to different locations, to renumber portions of your program, or to merge in some of your favorite routines. Now comes the test of using a good utility program.

You did not load and run the A/S -R/N & M Program prior to starting work on your program. Now what?

Save your program, load and run the A/S - R/N & M program, now load back in the program you were working on and you are ready to go again. Meanwhile, your train of concentration has been broken on what you were originally doing.

There is a better way; at least I think there is. If we plan ahead a little bit.

If the A/S - R/N & M program is set up as a 'B' file then when it is needed it can be 'BLOADED' into memory while our program that is being worked on stays in memory and undisturbed.

Here's the procedure in setting up an A/S - R/N & M 'B' file. The next time you boot a disk check to see what HIMEM: is set for right after the disk is booted. This is found by doing the following from the keyboard.

Print PEEK (115) +256 * Peek (116) (C/R).
(On a 48K HIMEM: 38400 - on a 32K HIMEM: 22016.) The next thing to do after recording your system HIMEM: is to load and run that outstanding renumber and merge program that APPLE Computer gave you on your master DOS 3.2 diskette. When the A/S prompt character RETURNS it means that the Renumber program has been put into a little corner someplace in your computer's memory, ready for your beck and call.

Actually where it resides in memory is right under your systems previous HIMEM: which was set when you first booted (this is the number you first recorded).
HIMEM: has now been changed by the renumber program. Let's record the new HIMEM: Again, from the keyboard.

Print PEEK (115) +256*PEEK (116) (C/R)

(On a 48K HIMEM: 36352 - on a 32K HIMEM: 19968).

We now have two numbers which we recorded. Subtract the smaller from the larger, this should equal 2048.

O.K., lets put the renumber program into a 'B' file on disk. From the keyboard:

BSAVE A/S-R/N-M, A (your 2nd HIMEM: number you recorded), L 2048 For a 48K this would look like BSAVE A/S-R/N-M, A36352, L2048 For a 32K BSAVE A/S-R/N-M, A36352, L2048 O.K., the 'B' file for the renumber program is all set.

Now, lets assume you are merrily programming away and the renumber program is not in memory.

The need occurs for renumbering, a merge or a hold. The newly created A/S-R/N-M 'B' file can now be 'BLOAD'ed in without disturbing your existing program. From the keyboard-BLOAD A/S-R/N-M (C/R).Once the 'B' file is loaded in, there are a few instructions that must be issued to your computer so that it knows the A/S-R/N-M program is in memory and where it is when it is needed. From the keyboard enter the following instructions;

For A 48K System:

```
HIMEM: 36352 (C/R)
POKE 1013,76 (C/R)
POKE 1014,0 (C/R)
POKE 1015,142 (C/R)
```

For A 32K System:

```
HIMEM: 19968 (C/R)
POKE 1013,76 (C/R)
POKE 1014,00 (C/R)
POKE 1015,78 (C/R)
```

O.K., that's it. You are all set to

use the Renumber program. As you will note, your existing program is still in memory and undisturbed. What the first instruction did was reset your system's HIMEM: below the A/S-R/N-M program that you just BLOADED in. This is required for when you use the hold feature of the program. The last three POKE instructions tell the ampersand character "&",which you use when using the renumber program, where to find the A/S-R/N-M program in your system. (see Applesoft manual p.123)

All the operating commands and formats that are used for the renumber program are valid and are used in the same manner. To free up the 2K of memory the A/S-R/N-M program is occupying, do a HIMEM: 38400 for a 48K system or a HIMEM: 22016 for a 32K system.

Now that you have come this far the ideal thing to do is set up a 'T' (text) file and let your disk 'exec' the whole procedure into your APPLE.

The program to write a text file would look like the following;

```
]LIST

10D$ = CHR$(4): REM CTRL D

20 PRINT D$; "OPEN RENUMBER-
MERGE"

30 PRINT D$; "WRITE RENUMBER-
MERGE"

40 PRINT "BLOAD A/S-R/N-M"

50 PRINT "HIMEM:36352:" REM for
32K system use 19968

60 PRINT "POKE 1013,76"

70 PRINT "POKE 1014,0"

80 PRINT "POKE 1015,142": REM
FOR 32K SYSTEM USE 78 IN PLACE
OF 142

90 PRINT D$; "CLOSE RENUMBER-
MERGE"

100 END
```

After the above program is run, a text file, named Renumber-Merge, will be created. Make sure this 'T' file is on the same diskette as your 'B' file A/S-R/N-M.

Now, whenever the renumber program is required all you have to do is type in EXEC Renumber-Merge.

μ

Frank Chipchase is presently employed as Chief Engineer for International Multifoods Corp. Until he purchased an APPLE computer, which was approximately one year ago, he had no expierence or contact with computers other than a programmable calculator. Although he purchased a computer for the pure fasination and challenge it would present, he has recently written a utility program for the APPLE computer which is presently being marketed for sale.

Variable Lister

This nifty little program will extract the variable names from your BASIC program, sort them, and list them alphabetically.

Ray Cadmus
600 W. Lee
Moberly, MO. 65270

I noticed a comment in a recent article bemoaning the lack of a BASIC X-REF FOR THE Apple. Perhaps this little program will help to fill that void.

The program is crude. It just scans the program area picking up words not in quotes and assumes them to be variable names. This approach also gets words from remark statements etc. Its one virtue is that it works.

Several changes could be made easily to enhance its operation. Turn off scanning after a REM token. Change the bubble sort to a Shell sort. (I use & sort myself). Set up an EXEC file to auto-append and execute, and so on. The most desireable change, of course, would be to rewrite in assembler. I plan to do that when time permits.

To use the program, simply append it to the one you wish to analyze and RUN 60,000. I use APPLE's renumber/append utility, but several have been published in MICRO and elsewhere. Another approach would be to list the program to a text file and the EXEC it into your program. μ

Ray Cadmus has been in data processing since the late 50's and programming since the early 60's. Most of his work has been with business applications on large scale IBM equipment. He started programming microcomputers because that would give him the opportunity to write what he wanted, rather than what business pressure dictated. Now, though he still works with micros for fun, he is expanding his consulting activities into the area of Small Business Computers and hopes to someday make that his primary occupation.

```
]LIST

60000  REM --VARIABLE LIST--6/7/79
60001  HOME
60002  PRINT "VARIABLE LISTER - RAY CADMUS": PRINT : PRINT
60003  PRINT "EXTRACTS AND PRINTS BASIC VARIABLES": PRINT : PRINT
60004  PRINT "AFTER THE FIRST PASS THRU"
60005  PRINT "PROGRAM - SORTS - THEN LISTS"
60006  PRINT "ALPHABETTICALLY"
60007  FOR D = 1 TO 1999: NEXT
60008  HOME
60009 LL = 59999: REM  ** HIGHEST LINE TO EXAMINE
60010 PD$ = "          "
60011  DIM T$(500),T(500)
60012  REM *****FIND FIRST LINE****
60013 NL = 2049
60014 P = 2049
60015 CL = NL
60016 NL =  PEEK (P) + ( PEEK (P + 1) * 256)
60017  IF NL = 0 THEN 60053
60018 P = P + 2
60019 LN =  PEEK (P) + ( PEEK (P + 1) * 256)
60020  IF LN > LL THEN 60053
60021 LAB$ = ""
60022  GOSUB 60032: REM  GET NEXT ALPHA
60023 LAB$ = LAB$ +  CHR$ (CH)
60024  GOSUB 60041: REM  GET NEXT CHAR
60025  IF CH > 64 AND CH < 91 THEN 60023
60026  IF CH > 47 AND CH < 58 THEN 60023
60027  IF CH = 36 THEN 60023
60028  IF CH = 37 THEN 60023
60029  GOSUB 60046
60030  GOTO 60021
60031  REM ****GET ALPHA CHAR
60032  GOSUB 60041
60033  IF CH = 34 THEN  GOSUB 60036
60034  IF CH < 65 OR CH > 90 THEN 60032
60035  RETURN
60036  GOSUB 60041
60037  IF CH <  > 34 THEN 60036
60038  GOSUB 60041
60039  RETURN
60040  REM  ******GET NEXT CHAR
60041 P = P + 1
60042  IF P = NL THEN  POP : POP : GOTO 60015
60043 CH =  PEEK (P)
60044  RETURN
60045  REM  ***** STORE LABEL ****
60046  PRINT LN;" ";LAB$,
60047 X = X + 1
60048 T$ = LAB$
60049 LN$ =  STR$ (LN)
60050 T$ =  LEFT$ (T$ + PD$,10)
60051 T$(X) = T$ + LN$
60052  RETURN
60053 : REM   ****** SORT RTN *****
60054  PRINT : PRINT : PRINT "SORTING--WAIT": PRINT
60055  FOR A = 1 TO X - 1
60056  FOR B = A + 1 TO X
60057  IF T$(A) < T$(B) THEN 60061
60058 HH$ = T$(A)
60059 T$(A) = T$(B)
60060 T$(B) = HH$
60061  NEXT B,A
60062  REM  *** LIST VAR TABLE ****
60063  HOME
60064  FOR C = 1 TO X
60065  PRINT T$(C)
60066  NEXT C
60067  PRINT
60068  INPUT "LIST AGAIN SLOWLY? ";Z$
60069  IF Z$ = "Y" THEN  SPEED= 150: GOTO 60064
60070  SPEED= 255
60071  END
```

MICRO Dealer Update

MICRO is preparing to publish another update on Dealers. There is still time to get your company listed. Simply send in the requested information and you too will be listed in our zip coded list. This is a service for our readers so that they may locate dealers in their area. Send this information to:

MICRO Dealer Update
P. O. Box 6502
Chelmsford, MA 01824

Dealer Information Form

Company Name:..............................
Address:..
City:..
State:........................Zip.................
Phone:.....................Hours...............
6502 Contact:..................................
6502 Products/Services:.......................
....

Additions to Tiny Pilot

These additions to Tiny Pilot include code to input a numeric variable, generate a random number, and call a machine language subroutine. A complete sample Pilot program is included.

Bob Applegate
Box 148
Bordentown, NJ 08505

Nicholas Vrtis' Tiny PILOT is a neat way to move up to a high level language, but it does have some drawbacks. One of the biggest problems is the lack of a method to input into a variable from running program. All values must be preset by a C: command. This can be a real hassle for some applications.

Another useful addition would be a machine language subroutine call. It would allow you to write programs using functions that standard Pilot doesn't have, like having a beeper rather than a "?" for a prompt. Or maybe comparing the contents of two variables and setting a flag to indicate which is larger.

One more function that could be added is a random number generator. Some games (until my KIM takes over the world, I'll resort to playing games on it!), such as HI-LO and CRAPS, can be played only if a random value can be created. If any of these problems bother you, then read on!

These routines will solve all of these problems. Before I start detailing them, realize that they will take away from memory space for the source (in Pilot). This will be no problem if your system has extra memory, but my 2K is filled really fast with a long program! Don't use a lot of remarks and long strings to conserve space.

Let me start by describing what modifications are needed to Tiny PILOT for these programs to work. Make the following corrections:

```
027E   4C   16   05
0281   EA
```

That just tells the interpreter to try to match the current command with our new ones before it checks its own. The instructions that we just wiped out are replaced at 0516. Correct the following:

```
048C   A9   06
```

That tells the interpreter that the Tiny PILOT source begins at page 6, not page 5. Addresses 04FA ot 0515 are just relocated versions of the subroutines previously described by me (MICRO, 21:41). If your system doesn't need them, relocate the rest of the program to 04FA. If you will be using them, remember to correct all the I/0 calls in the Pilot interpreter. Here are the new instructions:

I:x Input a positive number into variable x (can be any from A to Z). Prints a "?" as a prompt.

P:x Puts a random number into variable x (can be any from A to Z). The number will be in the range 0 to 99.

L:x Calls machine language subroutine x (can be any name from A to Z). The starting address of the subroutine is stored in the following table:

Name	Zero page address	
A	A0,	A1
B	A2,	A3
C	A4,	A5
.	.	
.	.	
.	.	
Y	D0,	D1
Z	D2,	D3

Here's how they work. 0516 to 0519 only replace what we destroyed at 027E to 0281. The first four instructions see if the next command is an L: command. If not, it jumps to 0531 for the next command. If it is the right one, it jumps to the subroutine at 0494 to get the index for the label name. Then it uses the index to get the starting address of the subroutine from the table (low-order first). Then it puts the values at the appropriate locations (052C, 052D) and makes a jump to the subroutine. This routine can't be PROMmed.

There is probably a better way to execute that jump, but this way is easy, and it works. Finally it jumps back to 0279.

I can't take credit for the random number generator (053A to 054B). It is a slightly modified version of the one presented by Jim Butterfield on page 172 of *The First Book of KIM*. I suggest that you look there for the theory behind it. Addresses 0531 to 0534 just check to see if we are executing the correct command. A call is made to 0494 for the index. The X register is stored for future use at

008D. Then the random number is produced. The result is in the A register. X is loaded again; then the value of A is stored in the proper variable. It finished by jumping to 0279

All that is left now is the I: command. If it's not an I, the program jumps to 0591. The next five instructions output a prompt character ("?") and clear the temporary work area (00DA, 00DB). Then it gets the ASCII input. If it is a CR, it jumps ahead to 0580. Otherwise, it subtracts $30 to get a decimal number. Next, it rolls 00DA and 00DB four places to the left, to make room for the new digit. The value of the A register is added to 00DA to achieve the new number. The program jumps back to 0564 to get the next character.

Once a CR input, the program goes to 0580. Then it jumps to 0494 for the index. The contents of 00DA and 00DB are stored at the proper variable. Then the program outputs a CR and LF, and finally jumps back to 0279.

If the command didn't match any of those, the program goes back to 0282, where it looks through the standard Pilot instructions. Additional commands can be added from 0591 and up. The A register will already contain the command character, so just use a CMP instruction to see if it is the one you want. The Y register already points to the character after the ":", so just use a B1 97 to load it into the A register. The last instruction should be 4C 79 02. The very last instruction after your additional routines must be 4C 82 02.

I hope that these new commands will increase the use of Tiny PILOT. It is really a good language, considering its small size. I have included some sample Tiny PILOT programs to demonstrate what it can do. μ

Bob Applegate is seventeen years old, an 11th grade student. He has been accepted to a local college where he plans to major in computer science. He has been working with computers for about four years, starting with BASIC, at Princeton University.

His one-year-old KIM is about to be upgraded to 16K, with OSI BASIC-in-ROM.

New Pilot Commands

```
0516-  85 87     STA  $87     CLEAR HIGH BIT FOR EDITOR
0518-  C8        INY          POINT TO ":"
0519-  C8        INY          AND THE NEXT CHARACTER
051A-  C9 4C     CMP  #$4C    IS IT THE L COMMAND
051C-  D0 13     BNE  $0531   NO, GO TO 0531
051E-  20 94 04  JSR  $0494   COMPUTE THE INDEX
0521-  B5 A0     LDA  $A0,X   GET THE HIGH-ORDER BYTE FROM TABLE
0523-  8D 2C 05  STA  $052C   PUT IT BEHIND THE JSR
0526-  B5 A1     LDA  $A1,X   GET THE LOW ORDER BYTE
0528-  8D 2D 05  STA  $052D   PUT IT BEHIND THE JSR
052B-  20 00 00  JSR  $0000   EXECUTE THE SUBROUTINE
052E-  4C 79 02  JMP  $0279   ALL DONE, RETURN TO PILOT
0531-  C9 50     CMP  #$50    IS IT THE P COMMAND
0533-  D0 20     BNE  $0555   NOPE, GO TO 0555
0535-  20 94 04  JSR  $0494   COMPUTE THE INDEX
0538-  86 8D     STX  $8D     STORE THE INDEX FOR NOW
053A-  F8        SED          DECIMAL NUMBERS ONLY, PLEASE
053B-  38        SEC          CARRY ADDS VALUE 1
053C-  A5 D5     LDA  $D5     LAST VALUE
053E-  65 D8     ADC  $D8     ADD B+CARRY
0540-  65 D9     ADC  $D9     ADD C
0542-  85 D4     STA  $D4     NEW NUMBER
0544-  A2 04     LDX  #$04    MOVE 5 NUMBERS
0546-  B5 D4     LDA  $D4,X   GET FIRST NUMBER
0548-  95 D5     STA  $D5,X   MOVE OVER 1
054A-  CA        DEX          NEXT NUMBER
054B-  10 F9     BPL  $0546   ALL MOVED?
054D-  D8        CLD          EVERYTHING BACK YO NORMAL
054E-  A6 8D     LDX  $8D     PICK-UP THE INDEX
0550-  95 54     STA  $54,X   STORE THE NUMBER AT VARIABLE
0552-  4C 79 02  JMP  $0279   BACK TO PILOT
0555-  C9 49     CMP  #$49    SEE IF IT'S THE I COMMAND
0557-  D0 38     BNE  $0591   GO TO 0591 IF NOT
0559-  A9 3F     LDA  #$3F    GET A "?"
055B-  20 02 05  JSR  $0502   OUTPUT IT SAS A PROMPT
055E-  A9 00     LDA  #$00    CLEAR THINGS OUT
0560-  85 DA     STA  $DA     ESPECIALLY THIS TEMPORARY VARIABLE
0562-  85 DB     STA  $DB     DITTO
0564-  20 FA 04  JSR  $04FA   GET INPUT
0567-  C9 0D     CMP  #$0D    CR?
0569-  F0 15     BEQ  $0580   YES, GO TO 0580
056B-  38        SEC          GET READY TO SUBTRACT
056C-  E9 30     SBC  #$30    TURN ASCII INTO BCD
056E-  A2 04     LDX  #$04    GET READY TO MULTIPLY BY 10
0570-  18        CMC          CLEAR THINGS FIRST
0571-  06 DB     ASL  $DB     MULTIPLY
0573-  26 DA     ROL  $DA     MULTIPLY
0575-  CA        DEX          AGAIN?
0576-  D0 F8     BNE  $0570   YES, THEN 0570
0578-  18        CMC          CLEAR THINGS UP AGAIN
0579-  65 DB     ADC  $DB     ADD THE NEW DIGIT
057B-  85 DB     STA  $DB     STORE THE ANSWER
057D-  4C 64 05  JMP  $0564   DO IT ALL OVER AGAIN
0580-  20 94 04  JSR  $0494   GET THE INDEX
0583-  A5 DA     LDA  $DA     GET FIRST PART OF ANSWER
0585-  95 53     STA  $53,X   STORE IT AT VARIABLE
0587-  A5 DB     LDA  $DB     GET THE NEXT PART
0589-  95 54     STA  $54,X   AND STORE THAT
058B-  20 0A 05  JSR  $050A   START A NEW LINE
058E-  4C 79 02  JMP  $0279   ALL DONE, RETURN TO PILOT
0591-  4C 82 02  JMP  $0282   NOT A NEW COMMAND, CHECK OLD ONES
```

Some Relocated Subroutines
(See MICRO 21:41)

```
KIM
07DC 20 G
04FA-   84 EE      STY  $EE
04FC-   20 5A 1E   JSR  $1E5A
04FF-   A4 EE      LDY  $EE
0501-   60         RTS
0502-   84 EE      STY  $EE
0504-   20 A0 1E   JSR  $1EA0
0507-   A4 EE      LDY  $EE
0509-   60         RTS
050A-   86 ED      STX  $ED
050C-   84 EE      STY  $EE
050E-   20 2F 1E   JSR  $1E2F
0511-   A6 ED      LDX  $ED
0513-   A4 EE      LDY  $EE
0515-   60         RTS
```

EXAMPLE

```
>R:BLACK-MATCH
>T:What is your name
>?:
>T:Hello $?, you and the computer take turns by removing 1,
>T:2, or 3 matches from the pile of 21 matches.  Whoever takes
>T:the last match loses.  Good luck!
>*SC:A=21
>C:B=121
>*HT:
>T:Enter your move
>I:C
>C:D=4-C
>T:My move is $D
>C:A=A-4
>C:B=B-4
>T:The new total is $A
>C:$=B
>M:101
>NJ:H
>T:Your next move must be 1 $?, so I win!
>T:Do you want to try again?
>A:
>M:Y,y
>YJ:S
>T:Don't feel bad $?, this program
                        can only win.
>S:
>
```

BLACK MATCH: The first thing we do is get the player's name and put it into the name field. A and B contain the number of matches left, but B has 100 added to it. Tiny Pilot has problems with its match statement , so it is necessary to have both A and B. We get the player's guess and subtract it from four to get the computer's move. If B equals 101, meaning only one match is left, the computer claim its victory, otherwise it gers the players's new move.

```
What is your name
?BOB
Hello BOB, you and the computer
take turns by removing 1, 2,
or 3 matches from the pile of 21
matches.  Whoever takes the
last match loses.  Good luck!

Enter your move
?2
My move is 2
There are 17 matches left

Enter your move
?1
My move is 3
There are 13 matches left

Enter your move
?2
My move is 2
There are 9 matches left

Enter your move
?3
My move is 1
There are 5 matches left

Enter your move
?1
My move is 3
There are 1 matches left
```

```
Your next move must be 1,
so I win!
Want to try again?
?yes

Enter your move
?2
My move is 2
There are 17 matches left

Enter your move
?4
My move is 0
There are 13 matches left

Enter your move
?1
My move is 3
There are 9 matches left

Enter your move
?76
My move is -72
There are 5 matches left

Enter your move
?2
My move is 2
There are 1 matches left
Your next move must be 1, so I win!

Want to try again?
?no thanks!
Don't feel bad, BOB, this
      program can only win!
```

HDE inc.

BOX 120
ALLAMUCHY, NJ 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

ANNOUNCING:
HDE DISK USER LIBRARY

The HDE Disk User Library has been established to facilitate the free exchange of general and specific routines for use on HDE disk based systems.

All library programs are 'public domain', and are available for a modest copying charge to defray expenses. In most cases, the source and object programs together with instructions for use are provided on diskettes submitted by the requestor.

HDE disk users are encouraged to submit programs for free distribution. A list of programs currently available may be obtained by sending a S.A.S.E. to:

Progressive Computer Software
405 Corbin Road
York, PA 17403
717-845-4954

JOHNSON COMPUTER
Box 523, Medina, Ohio 44256
(216) 725-4560

FALK-BAKER ASSOCIATES
382 Franklin Ave., Nutley, NJ 07110
(201) 661-2430

PLAINSMAN MICROSYSTEMS
Box 1712, Auburn, Ala. 36830
(800) 633-8274

PERRY PERIPHERALS
P.O. Box 924, Miller Place, NY 11764
(516) 744-6462

LUX ASSOCIATES
P.O. Box 315
Chico, CA 95927
(916) 895-8751 (for SYM Versions)

PROGRESSIVE COMPUTER SOFTWARE
405 Corbin Rd., York, PA 17403
(717) 845-4954

AB Computers
115 East Stump Road
Montgomeryville, PA 18936
(215) 699-8386

Frustrated? One-board Micro-P's too expensive, Insufficient I/O line for full control, General purpose Micro an overkill, Fixed power supply requirements, costly development systems, if available, lack of USER flexibility?

The MICROsport MicroComputer (MMC), in addition to being a complete microcomputer on a 4½" X 6" pc board, is the nucleus of a full hardware/software development system. Software can be developed for the MMC using any 6502-based computer. The In-Circuit Emulator (ICE) permits full MMC software/hardware debugging, then adding the EPROM Programmer Adaptor, any single +5V EPROM, such as the Intel-type 2716 or 2758 can be programmed without additional equipment. The MMC is the ideal dedicated controller for use in control/monitor systems, laboratory experiments, timing, intelligent interfaces, security systems, and other applications requiring a low cost controller.

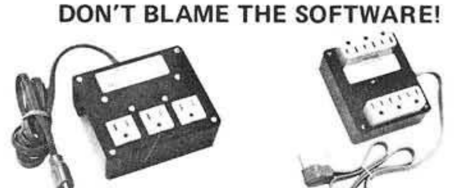
Features

CPU: MPS6503, **RAM:** 1 Kbytes, **EPROM:** Socket for Intel-type 2716 or 2758. (User supplied), **I/O:** 2 MPS6522 VIAs: 32 I/O lines, 8 edge detector/control lines, 4-16 bit counters, 8user-defined pins. **Serial:** 20 mA loop on-board, **Interrupts:** Power-on and manual reset, non-maskable and maskable, **Power:** +5V regulated or 9—20V regulated AC or DC, **Also:** ICs sockets, LED power on indicator, CMOS-RAM w/battery back-up optional.
Options: In-Circuit Emulator; EPROM Programming Adaptor; MMC Development Model MMC/03D with zero insertion force sockets (3); EPROM Programming Services; and application software development.

Prices: Kits from $89.00 Application units from $119.00
MMC/03S: Complete development system including MMC/03D; MMC/03ICE; MMC/03EPA; and software $250.00 (offer good until Dec. 31, 1980). Checks or money orders; No C.O.D..'s. For further info write or call:

R.J. Brachman Associates, Inc.
P.O. Box 1077
Havertown, PA 19083
(215) 622-5495

DISK DRIVE WOES? PRINTER INTERACTION? MEMORY LOSS? ERRATIC OPERATION? DON'T BLAME THE SOFTWARE!



ISO-1 ISO-2

Power Line Spikes, Surges & Hash could be the culprit! Floppies, printers, memory & processor often interact! Our unique ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

- ***ISOLATOR (ISO-1A) 3 filter isolated 3-prong sockets; integral Surge/Spike Suppression; 1875 W Maximum load, 1 KW load any socket $56.95**
- ***ISOLATOR (ISO-2) 2 filter isolated 3-prong socket banks; (6 sockets total); integral Spike/Surge Suppression; 1875 W Max load, 1 KW either bank $56.95**
- ***SUPER ISOLATOR (ISO-3), similar to ISO-1A except double filtering & Suppression $85.95**
- ***ISOLATOR (ISO-4), similar to ISO-1A except unit has 6 individually filtered sockets $96.95**
- ***ISOLATOR (ISO-5), similar to ISO-2 except unit has 3 socket banks, 9 sockets total . . . $79.95**
- ***CIRCUIT BREAKER, any model (add-CB) Add $ 7.00**
- ***CKT BRKR/SWITCH/PILOT any model (-CBS) . . . . Add $14.00**

PHONE ORDERS 1-617-655-1532

ESP Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760

Dept. MI

MICRO Club Circuit

Keep an eye on what MICRO is saying about your club. Let us know if anything has changed. We are endeavoring to print what we have and there is a substantial amount waiting to be printed. But, all will be presented eventually. If your club is not listed this month, it will be next. Or haven't you registered your club with MICRO? If not, please send for our easy registration form. This will then qualify your club for one year of MICRO, FREE! Your club's library should not be without every issue of MICRO.

What can MICRO do for you? How can we better serve your interests? Jot down any suggestions you have. Ask your members and let us know. Address any suggestion, comments, up-dates, registrations, etc., to:

MICRO Club Circuit
P.O. Box 6502
Chelmsford, MA 01824

Dallas Apple Corps

Meets on the second Saturday of every month from 2 to 5 pm at the Greenhill School. (14255 Midway Road, Dallas, TX) Membership at 250. Monthly newsletter, Apple-Gram, $12.00 per year. Contact Club President:

Bob Sander-Cederlof
P.O.Box 5537
Richardson, TX 75080

"For Apple owners, to share what we have learned and to promote enjoyment of personal computing."

PTC

This West-German club consists of approximately 300 members who meet at Gutenbergstr. 20, D-6052 Muehlheim, West Germany. They publish a bi-monthly informational newsletter. For more information contact:

Dietmar Svertitt
Gutenbergstr. 20
D-6502 Muchlheim 1,
West Germany

"Exchanging programs and information."

A.B.A.C.U.S.

Meets on the second Monday of the month with a membership of 300. Ed Avelar is club President. Contact Larry Danielson for more information:

5302 Camino Alta Mira
Castro Valley, CA 94546

"Apple owners in one group to share ideas. Apple Bay Area Computer User Society."

Micro Computer Club

This club meets on the first and third Thursday of each month. The membership is continuing to grow, is at 55 now. Alfredo Buzali, President. Contact:

Alfredo Buzali
fte de Quijate 5
Tecamuchalco, Mexico 10

"The only computer club in Mexico. Publishes a bulletin describing the past meeting."

6502 Club/Copenhagen

Meets once a month on various days. Membership about 100. Contact:

Erik Skovgaard
Nordlundsvej 10
DK-2650 Hvidovre
Denmark

"General exchange of experience, review of systems, setup of a software library. Collective Administration. Several subdivisions of interest groups."

Forth Interest Group

Meets on the fourth Saturday at Noon. Membership over 1200. Club address:

P.O.Box 1105
San Carlos, CA 94070

Contact person:

Jim Flournay Ancon
17370 Hawking Lane
Morgan Hill, CA 95037

"Forth Dimensions, Publication at $12.00 per year, back issues $6.00."

Philadelphia Apple Club

Meets on the third Saturday of each month ath the Science Building at LaSalle College. Neil D. Lipson, President. Write to him at:

29 S. New Ardmore Avenue
Broomall, PA 19008

"Apple Computer Programming and Hardware Discussion."

Apple II Amateur Radio Computer Network

"Apple owners who are amateur radio operators are checking into the Apple II computer network. The 'net' meets every Sunday evening on 14.329 MHZ in the 20-meter band. Communication is by upper side-band voice transmission. Actual time is 0100 UTC every Monday. (That is 6pm Pacific time Sunday night or 9pm Eastern time Sunday night.) Use of the Apple II in amateur radio applications is the prime topic, but for new ham radio users, there is help with specific programming and hardware questions. Net control is Jim Hassler, WB7TRQ in Cheyenne, Wyoming."

De Vlaamse Minicomputerclub vzw

Here is a club in Belgium. Write to Cornelis Bergmans, Club President for more information, at:

Lambrechtshoekenlaan
171 b6
B-2060 Merksem
Belgium

Nth Precision Add & Subtract With Adjusted Processor Status

Here is a general purpose utility which can be used with 6502 programs which require addition and subtraction.

Lawrence R. Golla
21630 Mallard Ct.
Brookfield, WI. 53065

There must be hundreds of add and subtract routines of varying precision, and with the ability to utilize the "Branch" instructions after the call to the subroutines, but of the 6502 users there need by only one routine no matter what precision is required. These subroutines eliminate the need for two or more add/ subtract subroutines to handle the appropriate precision or save execution time of addition/subtraction of small precision numbers that utilize subroutines of greater precision, i.e. adding 16-bit numbers with a 24- bit add subroutine.

Before calling the addition or subtraction subroutines, five locations of page zero memory must be preset. Two bytes are used to store the address of the addend or subtrahend, two bytes are used to store the address of the augand or minuend, and one byte to store the degree of precision.

Since the subroutines use the indirect zero page mode, the addresses stored in page zero must be stored as least signficant byte first followed by the most significant byte. Indirect addressing mode is used by the subroutines in place of storing data into page zero for two reasons: First, to prevent the necessity of moving data twice; into page zero and then out of page zero. Second, to conserve page zero memory especially when more than double precision is required. If data were stored directly into page zero, a 32-bit add would require eight bytes of page zero; twelve bytes of the program is modified to store the sum in another location. The subroutines replace the addend or subtrahend with the sum. If this is not desired, two more bytes of page zero memory are required to store the address of the sum. The program must be modified at locations $02E1 for addition and $0223 for subtraction to read 'STA (SUM), Y". "SUM" of course is equated to some page zero location.

The degree of precision stored in page zero is 1* "number of bytes of precision" -1 or,

0 = 1 byte precision
1 = 2 byte precision
2 = 3 byte precision, etc.

The Y register is loaded with this value to fetch and store the appropriate data byte(s).

In addition the accumulator, X, and Y registers are not destroyed, but the processor status reflects any changes to the N,C,Z, and V flags due to the result of the addition or subtraction. This allows the use of the branch instructions immediately following the call and utilizing the accumulator, X, and Y registers as preset prior to the call.

Locations $0228 to $0251 adjust the processor status for "N" and "Z" flags. The "C"and "V" flags were adjusted previously by the addition or subtraction. Each byte of the sum is exclusively OR'ed to adjust the "Z" flag but, if this results in setting the "N" flag, the program forces the "Z" flag to be reset, i.e. it is assumed that -0 is undefined but more importantly the program handles the problem of having a sum = $0080. The "N" flag is adjusted by exclusively OR'ing zero with the most significant byte of the sum.The status is then stored temporarily at location $0100 and retrieved oonly after the A, X, and Y registers are restored in order to avoid damaging the status.

A source listing followed by a disassembler listing equates locations $0010 and $0011 for the address of the addend or subtrahend; locations $0012 and $0013 for the address of the augand and minuend; and location $0014 for the value of the precision. The sum of the result is stored indirectly through locations $0200, and the subtract subroutine is located at location $0215.

```
         LDA  #01             ;Set precision for 16 bits
         STA  PREC
         LDA  #00             ;0300 and 0301 is the address
         STA  AEND            ;of the addend and sum
         LDA  #03             ;0302 & 0303 is the address
         STA  AEND+1          ;of the augand
         STA  AGAND+1
         LDA  #00             ;initially add 0 + 10
         STA  $0300           ;the constant 6 will be
         STA  $0301           ;subtracted from the sum
         STA  $0302           ;this minuend is at address
         STA  $0304           ;$0304 & 0305
         LDA  #$0A
         STA  $0303
LOOP     LDA  #02
         STA  AGAND
         LDA  #06             ;A & X reg. are pre loaded
         LDX  #04             ;before going to subroutine
         JSR  ADD
         BMI  OUT             ;test after the call
         STA  $0305           ;preloaded values made before
         STX  $12             ;call sets up minuend.
         JSR  SUB
         JMP  LOOP
         BRK                  ;exit here when sum is minus.
         BRK
```

To illustrate how to use the subroutines, a small program adds two numbers and subtracts a small number from the sum. The process is repeated until the sum is minus. The program can be expressed by the following formula;

$k_n = k_n.1 + 10 - 6, k_0 = 0$

The program shows how to set up the addresses and precision values; shows registers can be preset before a call to the add/subtract subroutines so they may be used after the call; and shows how to test andand branch immediately after the call from the subroutines.

A disassembled listing ORG'ed at $0266 follows. At the "break" location $0300 can be checked to see if it is indeed minus. This program takes a few seconds to run, which means the display will be blank until the "BRK" is encountered.

μ

```
; NTH PRECISION ADD/SUBTRACT
;
AEND    =     $10            ;ADDR'S OF ADDEND OR SUBTRAHEND
AGAND   =     $12            ;ADDR'S OF AUGAND OR MINUEND
PREC    =     $14            ;DEGREE OF PRECISION
;
        *=    $0200
ADD     PHA
        TYA
        PHA
        TXA
        PHA
        LDY   PREC
        CLC
        CLD
        CLV
LOOP1   LDA   (AEND),Y       ;ADD
        ADC   (AGAND),Y
        STA   (AEND),Y       ;REPLACE ADDEND WITH THE SUM
        DEY
        BPL   LOOP1          ;GET NEXT BYTE
        BMI   OUT            ;GO ADJUST FLAGS
SUB     PHA
        TYA
        PHA
        TXA
        PHA
        LDY   PREC
        CLD
        SEC
        CLV
LOOP3   LDA   (AEND),Y       ;SUBTRACT
        SBC   (AGAND),Y
        STA   (AEND),Y       ;REPLACE SUBTRAHEND WITH THE SUM
        DEY
        BPL   LOOP3          ;GET NEXT BYTE
```

```
OUT     LDY   PREC
        LDA   #0
LOOP2   EOR   (AEND),Y  ;ADJUST Z-FLAG
        PHP
        BMI   NZER      ;IF BIT 7=1 RESET
                                   Z-FLAG
LOOP4   DEY
        BMI   OUT1      ;IF NO MORE BYTES
                        , ADJUST N-FLAG
        PLP
        JMP   LOOP2
NZER    PLP
        ORA   #$01      ;FORCE Z-FLAG TO
                                    RESET
        PHP
        JMP   LOOP4     ;GET NEXT BYTE
OUT1    PLA             ;GET STATUS
        AND   #$7F      ;RESET N-FLAG
        STA   $0100     ;SAVE IT
        INY             ;GET MS OF SUM
        LDA   (AEND),Y
        EOR   #0        ;ADJUST N-FLAG
        PHP             ;PUT STATUS IN A
        PLA
        AND   #80       ;SAVE ONLY THE
                               N-FLAG
        ORA   $0100     ;MERGE WITH SAVED
                                 STATUS
        STA   $0100     ;SAVE IT FOR LATER
        PLA             ;GET A,X,AND Y-REG
        TAX
        PLA
        TAY
        PLA
        STA   $0101     ;TEMPORARILY SAVE A
        LDA   $0100     ;GET STATUS
        PHA             ;PUT INTO STACK
        LDA   $0101     ;GET A
        PLP             ;AND STATUS
        RTS
        BRK
```

AIM Dissassembler Listing

```
<K>*=0200
/65
 0200 48 PHA
 0201 98 TYA
 0202 48 PHA
 0203 8A TXA
 0204 48 PHA
 0205 A4 LDY 14
 0207 18 CLC
 0208 D8 CLD
 0209 B8 CLV
 020A B1 LDA (10),Y
 020C 71 ADC (12),Y
 020E 91 STA (10),Y
 0210 88 DEY
 0211 10 BPL 020A
 0213 30 BMI 0228
 0215 48 PHA
 0216 98 TYA
 0217 48 PHA
 0218 8A TXA
 0219 48 PHA
 021A A4 LDY 14
 021C D8 CLD
 021D 38 SEC
 021E B8 CLV
 021F B1 LDA (10),Y
 0221 F1 SBC (12),Y
 0223 91 STA (10),Y
 0225 88 DEY
 0226 10 BPL 021F
 0228 A4 LDY 14
 022A A9 LDA #00
 022C 51 EOR (10),Y
 022E 08 PHP
 022F 30 BMI 0238
 0231 88 DEY
 0232 30 BMI 023F
 0234 28 PLP
 0235 4C JMP 022C
 0238 28 PLP
 0239 09 ORA #01
 023B 08 PHP
 023C 4C JMP 0231
 023F 68 PLA
 0240 29 AND #7F
 0242 8D STA 0100
 0245 C8 INY
 0246 B1 LDA (10),Y
 0248 49 EOR #00
 024A 08 PHP
 024B 68 PLA
 024C 29 AND #80
 024E 0D ORA 0100
 0251 8D STA 0100
 0254 68 PLA
 0255 AA TAX
 0256 68 PLA
 0257 A8 TAY
 0258 68 PLA
 0259 8D STA 0101
 025C AD LDA 0100
 025F 48 PHA
 0260 AD LDA 0101
 0263 28 PLP
 0264 60 RTS
 0265 00 BRK
 0266 A9 LDA #01
 0268 85 STA 14
 026A A9 LDA #00
 026C 85 STA 10
 026E A9 LDA #03
 0270 85 STA 11
 0272 85 STA 13
 0274 A9 LDA #00
 0276 8D STA 0300
 0279 8D STA 0301
 027C 8D STA 0302
 027F 8D STA 0304
 0282 A9 LDA #0A
 0284 8D STA 0303
 0287 A9 LDA #02
 0289 85 STA 12
 028B A9 LDA #06
 028D A2 LDX #04
 028F 20 JSR 0200
 0292 30 BMI 029F
 0294 8D STA 0305
 0297 86 STX 12
 0299 20 JSR 0215
 029C 4C JMP 0287
 029F 00 BRK
 02A0 00 BRK
<
<
<

<*>=0266
<G>/
 02A0 00 BRK
<M>=0300 00 02 00 0A
```

MICROBE

Here is a corrected version of the program listing for my article "Expanding the SYM-1...Adding an ASCII Keyboard" which appeared on pages 5-7 in the February, 1980 issue of MICRO (Number 21). Somehow the hex locations column of this listing was not used for the article. The program is fully relocatable, but to do so the 'INIT' routine must refer to the addresses to which the GKEY and KSTAT have been relocated.

Typos corrected on final version including label 'DISP' change to WAIT2 at location 206 (minor), incorrect object code fixed at line 222 to '20 47 8A' from 'A5 F1'. (Mneumonics were correct.) Last was pointer to KSTAT at line 240 was '40' now '39' which is correct.

Robert A. Peck

```
0200   20 88 81   GKEY    JSR   SAVER     SAVE REGISTERS
0203   AD 01 A8           LDA   A801      GET PARALLEL ASCII
0206   F0 24              BEQ   WAIT2     UNLESS NONE,THEN BRANCH
0208   85 F1              STA   00F1      STORE IT A WHILE
020A   A9 10              LDA   #$10      DEBOUNCE CONSTANT
020C   85 EF              STA   00EF      DEBOUNCE
020E   C6 F0      WAIT1   DEC   00F0      SMALL LOOP
0210   D0 FC              BNE   WAIT1
0212   C6 EF              DEC   00EF      LARGE LOOP
0214   D0 F8              BNE   WAIT1
0216   20 03 89   SCANA   JSR   IJSCNV    SCAN DISPLAY(USE SCANVEC)
0219   2C 01 A8           BIT   A801      IS KEY STILL DOWN?
021C   30 F8              BMI   SCANA     WAIT FOR KEY RELEASE
021E   A5 F1              LDA   00F1      KEY UP, PROCESS KEY
0220   29 7F              AND   #$7F      STRIP KEY STROBE BIT
0222   20 47 8A           JSR   OUTCHR    SEND INTO DISBUF
0225   A5 F1              LDA   00F1      GET IT AGAIN
0227   29 7F              AND   #$7F      STRIP IT AGAIN
022A   4C B8 81           JMP   RESXAF    RETURN WITH ASCII IN A
022C   A9 10      WAIT2   LDA   #$10      IF NO KEY,
022E   85 EF              STA   00EF      SCAN DISPLAY
0230   20 03 89   SCANB   JSR   IJSCNV    THRU SCANVEC
0233   C6 EF              DEC   00EF      A NUMBER OF TIMES
0235   D0 F9              BNE   SCANB     THEN GO BACK
0237   F0 CA              BEQ   GKEY      AND LOOK AGAIN
0239   AD 01 A8   KSTAT   LDA   A801      READ ASCII INPORT
023C   0A                 ASLA            SHIFT MSB INTO CARRY
023D   60                 RTS             RET, CFLAG=1 IF KEY DN.

0240   20 86 8B   INIT    JSR   ACCESS    UNPROTECT SYSRAM
0243   A9 00              LDA   #00       MODIFY
0245   8D 61 A6           STA   A661      KEYBOARD
0248   A9 02              LDA   #02       INPUT
024A   8D 62 A6           STA   A662      VECTOR
024D   A9 39              LDA $39
024F   8D 67 A6           STA   A667      KEYPRESS
0252   A9 02              LDA   #02       STATUS
0254   8D 68 A6           STA   A668      VECTOR
0257   4C 03 80           JMP   WARM      WARM ENTRY,MONITOR
```

MICRO SCOPE

Number 1

BASIC Programmer's Toolkit™

1. Microcomputers which can use product: The Basic Programmer's Toolkit ROM is for any PET or CBM computer, except the new 8032.

2. System hardware requirements: One version comes on a small circuit board which plugs into the memory expansion port of original-model 4 & 8K PETs. The Basic 2.0 version is a single ROM chip which plugs into the expansion socket addressed at $B000. There are also special versions for owners of Skyles memory boards or Computhink disks. All versions require one cassette recorder for the 'Append' command.

3. System software requirements: Separate versions are available for Basic 1.0 & Basic 2.0 ('old' or 'new').

4. Product features: The features of the Toolkit are well known now, with over 10,000 sold. It works by adding commands to PET Basic. The commands added include: APPEND — joins 2 programs from cassette; AUTOMATIC LINE NUMBERING; RENUMBERING of Basic lines; DELETION of a range of lines; a HELP command — which lists and highlights in reverse field the character in a line which caused an error message; TRACE, which displays the last 6 line numbers executed in reverse field at the top right of the screen during a program run; STEP, which does the same, but goes to the next line only when you hit a key — or quickly when shift is held down; OFF, merely turns off trace and step; FIND, which finds every occurence of a token or characters in a program; and DUMP, which displays all non-array variables and their current values.

5. Product performance: All the commands work and work well. It is one of the very few uncrashable programs I own. In the 9 months I have had the Toolkit, I have never lost a byte or a minute due to any Toolkit malfunction. Having it in ROM is a great convenience. It is also completely compatible with DOS support 4.0 (the wedge).

6. Product quality: The quality of this program is excellent. It is effective, reliable, rapid and unobstructive in use.

7. Product limitations: There is only one known bug in the Toolkit. Once the step mode is left, to do a dump of variables for example, it is not possible to continue from where you stopped. This is unhandy. However, the same effect can be arranged by inserting stop statements and using trace instead. (Then use 'find' to remove the stop statements.)

The best-known problem is that Commodore decided to put both Word Pro I and Basic IV in the same ROM slot the Tool kit uses. There are switcher boards available for under $30, if you use Word Pro II. Basic IV will need its own Toolkit. Another minor gripe is that when the find or dump commands are directed to the PET printers, no carriage returns are sent, leaving the output squished together, 80 columns wide. A kill command, to remove the wedge into Basic would have been nice.

I'm sure the Toolkit slows Basic slightly, though I don't notice the difference in normal use. The other wished-for command is the change command in Commodore's 'Basic-Aid' program (not available for sale, but some users have it.) It allows users to replace a word or phrase everywhere in a program at once.

8. Product documentation: The instructions for the Toolkit are excellent. They are well written, usable and complete; they come in an attractive and durable manual.

9. Special user requirements: The only special requirement is to link the Toolkit with Basic each time PET is turned on, with 'sys11*4096'. Pushing the ROM in is simple enough for most users to do it themselves.

10. Price/Feature/Quality evaluation: I consider the Toolkit essential equiptment for all PET owners who write programs. It will quickly repay its cost in programming time saved. Even now, several months after its introduction, it has no real competition in features. (*Ed's Note:* Price: $50 for Basic 2.0 Version, others price depending on configuaration.

11. Additional comments: The Toolkit may be ordered through many computer stores, or directly from PAICS, at 430 Sherman Avenue, Palo Alto, CA. 94306. I also found them quite helpful on the phone. Their number is (415) 327-0125.

12. Reviewer: James Strasma, 120 W. King Street, Decatur, IL. 62521

Trade Mark of Nestar Systems, Inc.

APPLE II SOFTWARE FROM ON-LINE SYSTEMS

PADDLE-GRAPHICS:

The most powerful Hi-Res graphic development package available. Upper/lower case may be drawn on the Hi-Res screen in any size, direction or color. Pictures may be sketched and filled in with any of 21 colors (must be seen to believe!). A shape may be constructed automatically from any object appearing on the Hi-Res screen.

48K Applesoft/Machine Language
$39.95/disk

TABLET-GRAPHICS:

All of the capabilities of PADDLE-GRAPHICS extended for use with APPLE'S GRAPHICS TABLET.

48K Applesoft/Machine Language
$49.95/disk

HI-RES ADVENTURE #1:

Your APPLE computer becomes your eyes and ears as you enter a spooky old mansion in search of treasure. You direct it in English to move you through the house which is portrayed in full HI-RES GRAPHICS. You are in complete comtrol as you open caabinets, smash walls and manipulate any items you might find. Danger is ever present as you find your friends being murdered one by one. This game will definitely challenge your imagination. French version available upon request.

Machine Language for the Apple II or II plus 48K
$24.95/disk

SKEETSHOOT:

A real-time Hi-Res graphic simulation of the sport of skeetshooting. This game has all the sounds, action, and excitement of the real sport as well as the official firing positions. One to five players may compete.

Machine Language for the Apple II or II plus 48K
$19.95/disk

TRAPSHOOT:

One to five players compete at the sport of trapshooting. Fast action with many skill levels including adjustable shotgun spray make this game a true challenge.

Machine Language for the Apple II or II plus 48K
$14.95/disk

COMBINATION:

HI-RES ADVENTURE and SKEETSHOOT both on disk.

Machine Language for the Apple II or II plus 48K
$37.50/disk

Available at your local computer store.

ON-LINE SYSTEMS
772 N. HOLBROOK ST.
SIMI VALLEY, CA 93065
805-522-8772

Dealer Inquiries Invited

All orders shipped same day.

Software may be ordered directly by calling/writing: 805-522-8772. Orders may be: COD/CHECK/MASTERCHARGE/VISA. Add $1 shipping.

California residents add 6% sales tax.

Solar System Simulation with or without an APPLE II

Here is a fascinating program which combines the graphics of the Apple with the Laws of the Universe to make a super demonstration.

David A. Partyka
1707 N. Nantuckett Dr.
Lorain, OH. 44053

There are unlimited things to do with a micro that has high resolution graphics. Some of the more fascinating aspects are the simulation of objects around us. This article and program deals with the simulated motion of the first six planets of our solar system.

Each planet moves in an elliptical orbit of varying distance from the sun. The closer the orbit to the sun, the less time it takes that planet to complete its orbit. Mercury, the closest planet takes 88 days, while Saturn the farthest of the first six takes 29 years. Because the planets move in elliptical orbits, their distance from the sun and orbital speed is constantly changing. Using Johann Kepler's (1571,1630) second law of planetary motion "The line joining the planet to the sun sweeps out equal areas in equal time", we can calculate the time it takes the planet to travel from point W to point R (figure 1). As can be seen, the line RV joining the sun S to the planet R will vary in length as the planet travels around its orbit. Being at its minimum distance at W, the planet must travel faster for the line RV to sweep an equal area as when the planet is at its maximum distance Z.

To calculate the area SWR (figure 1) we use the formula

1.) $\text{Area} = \frac{ab}{2} \cdot (H - e \sin H)$.

Variable *a* being the length of the major axis, *b* the length of the minor axis, *e* the eccentricity of the ellipse *(c/a)* and H (figure 2), the angle in RADIANS from the center of the ellipse to point *q*. Point *q* being on a circle of radius *a*, intercepted by a perpindicular line form the major axis going thru point R to the circle.

By using Equation (1), we can calculate the number of days it takes the planet to travel any degree of angle from the area. By dividing the total area of the ellipse, (total area = pi ab), by the number of days to complete the orbit we have the area swept out per day. Rearranging equation (1), we get

2). $H - e \sin H = \frac{\text{area} * 2}{ab}$

and a problem. The term H-e sin H can't be simplified for the angle H because of the term sin H. Given the daily area we could still calculate the angle H by using a loop routine until we got the correct answer, but this would considerably slow the simulation down.

Instead I use the angle A (figure 1)

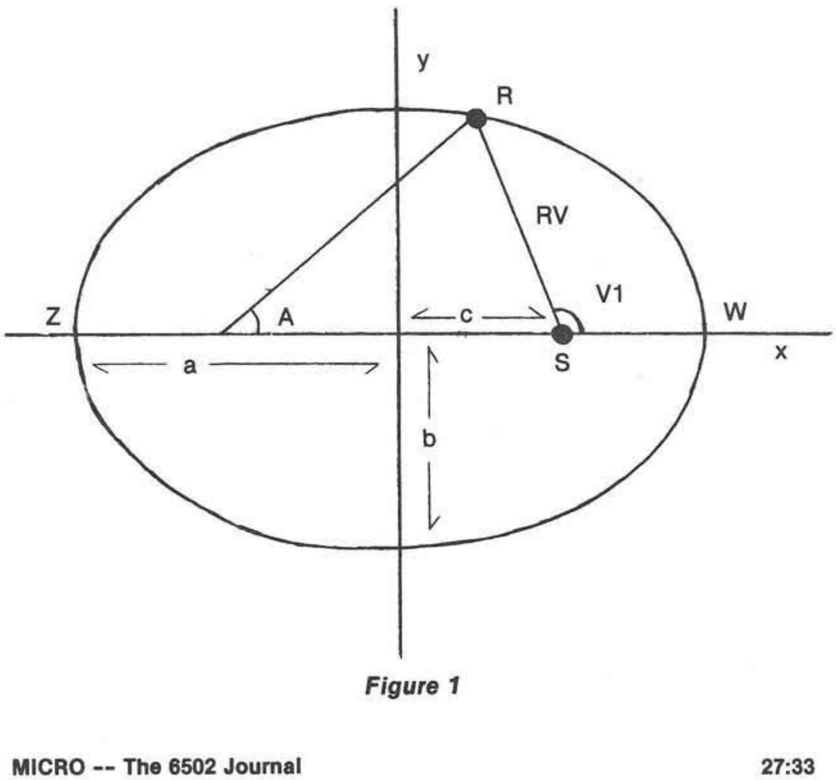


Figure 1

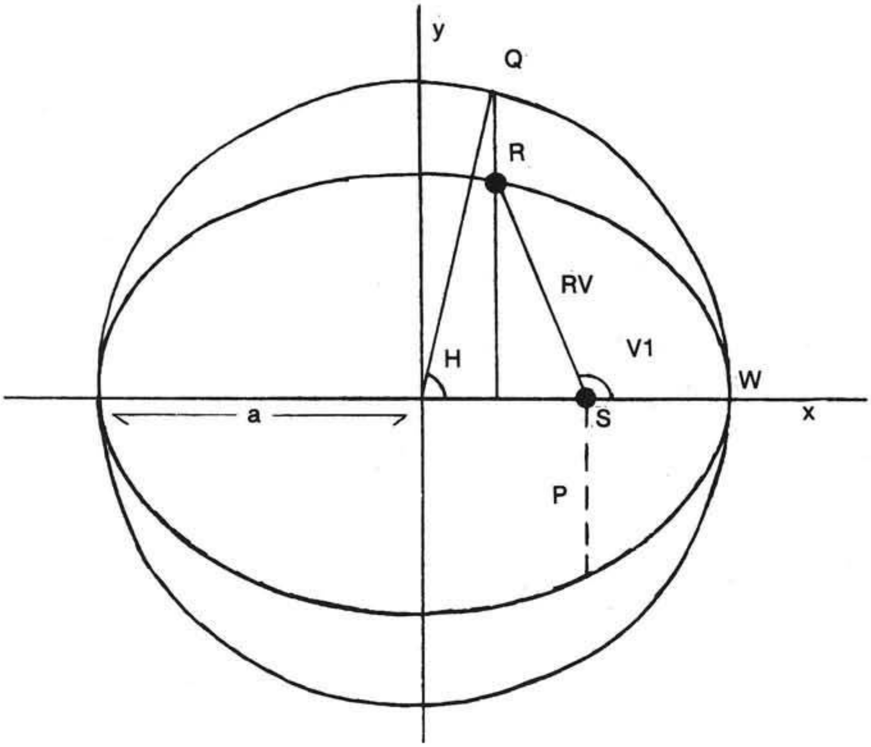


Figure 2

at the other focus of the ellipse. By dividing 360 degrees by the number of days to complete the orbit we get the number of degrees per day for angle A. Using the equation

$$3.)\ RV = 2 - (P/l + e(\cos(180 - A)))$$

we get the distance between the sun and the planet for each value of A. Using another equation

$$4.)\ \cos V1 = \frac{P - RV}{RV^{*}e}$$

we get the angle V1 that the planet lies in relation to the sun (figure 2). The value P in equation (4) being a perpendicular line from the focus to the ellipse and equal to $a(L-e^2)$. By increasing angle A at the daily rate we get the X,Y coordinates for each day and plot it on the screen.

Using angle A also causes a problem. Increasing angle A at a daily rate doesn't increase the area SWR (figure 1) at a daily rate. Even though there is an error, it isn't accumulative. The difference returns to zero at four points in the orbit, two points being at the minimum point W and the maximum point Z. The other two points vary with eccentricity but zero out before the ¼ position and after the ¾ position of its orbit. For Mercury, the fastest planet the error amounts to about .65 degrees and even less for the other planets. One more equation,

$$5).\ \cos H = \frac{a\text{-}RV}{ae}$$

is a link between equation (1) and equation (3) and can be used to calculate the error of using angle A.

Now that the calculations are out of the way, let me describe this program. To keep the program small I chose only the first six planets. If you want to add the other three planets it can be done with little trouble, see Listing 1. The planets are plotted in order from the sun, Mercury, Venus, Earth, Mars, Jupiter, then Saturn. You can choose any combination of planets to display, from one to all six. The planets are assigned scaling factors so its orbit will use the full plotting area when selected planets are used.

You can plot the position of the planets or planet for any day, i.e. July 8, 1980, or for any length of time

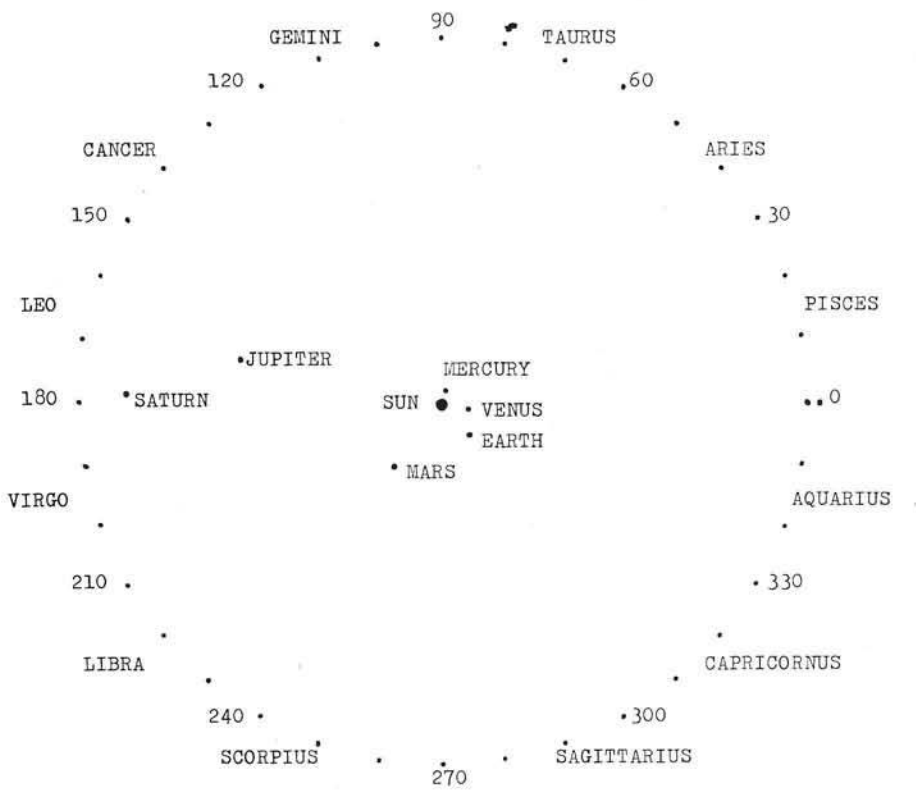


Figure 3: **This is an example of the display for all six planets for Aug 11, 1980 (224 days from Jan 0).**

from when you choose, ie. 100 days starting at Oct 3, 1980. You can plot any length of time with any amount of time between plots, ie. plot 900 days with 30 days beween plots. Then you can choose whether to plot single points, only one dot per planet, or continuous plots, each dot remains on the screen. Using single point plots it appears as if you are above the solar system looking down on the planets as they orbit the sun. With continuous plots you can see the orbit for the length of time you choose to plot with the amount of time between plots. When doing a plot, the first plot is always the date you choose, then it continues with what you requested. Figure 3 is an example of plotting all the planets for Aug 11,1980, 0 was the response for the number of days to plot with any number for days between plots. The constellation names, planet names, and degrees don't show on the actual display but are shown here for reference.

Figure 4 is an example of plotting the planets Mercury, Venus, and the Earth on May 29,1980 for 44 days with 4 days between plots. In this example May 29th was the first plot followed by the 11 plots for 44 days at 4 day intervals. Around the plotting area is a circle that has plots at 10 degree intervals with a double plot at the zero point. Use this to get the longitude of degrees that the planet lies in relation to the sun.
This program is set up for Jan. 0, 1980 of it you prefer Dec. 31, 1979. To change the reference date, just add the number of days difference from Jan. 0, 1980 to the values W, ie. W1, W2, W3, etc.

Some of the things you can do with this program are to determine the dates of superior conjunction, inferior conjunction, opposition, and greatest elongation. You can demonstrate the retrograde motion of the outer planets, whether a planet is a morning or evening object, or when two or more planets will appear close to each other in the sky. What else you can do depends on your knowledge of Astronomy, the program is simple so any additions or changes you make should be easy.

This program is written in floating point basic and uses the high resolution graphics subroutines. Since there are different types of APPLES, those with integer basic and others with floating point basic in ROM, how you use this program will depend on your system. Since I know my system I will describe what I had to do to it too.

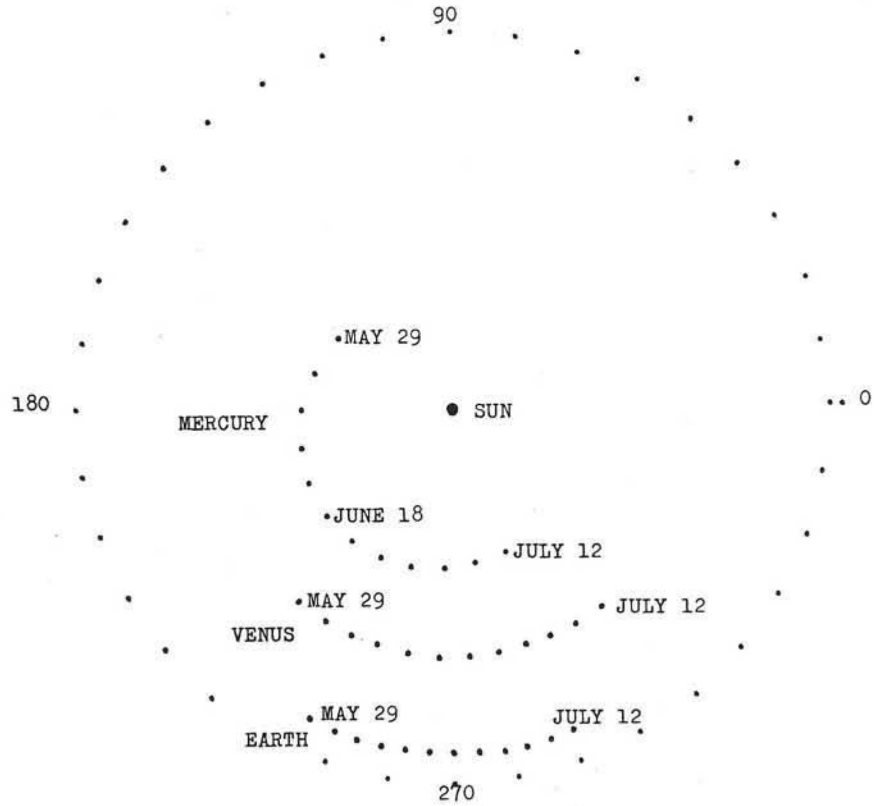


Figure 4: **This is a display of the planets Mercury, Venus, and the Earth. This example is for continuous plots starting May 29th (day 150), for 44 days with 4 days between plots.**

I have a 32K APPLE II with integer basic in ROM. My floating point basic is on cassette and loads from hex 800 to 29FF. My high resolution subroutines are also on cassette and load from hex C00 to FFF. Page 1 display for high res graphics hex 2000-4000 overlaps my floating point basic, so I have to use page 2 hex 4000-6000 for the display. Since floating point programs start loading at hex 2A00, large programs will overlap my page 2 display area so I had to change the program loading address from 2A00 to 6000. This is just beyond the page 2 display area.

On my system, after floating point basic is up and running I have to get into monitor and change hex locations 6B, 6D, 6F, 71 from 2A to 60. Then I have to change three bytes starting at hex 6000 to zeros, (*60000:00 00 00). If your floating point basic doesn't load from hex 800 to 29FF then look at the article "Applesoft Program Relocation" page 19:49 of the Dec. 1979 issue of Micro. If you don't have that issue, it looks like some floating point basics load from hex 800 to 2FFF. If this is yours then change hex locations 67 and 68, (*67:01 60). Then put zeros in the three bytes starting at 6000 like above. Typing 0G then RETURN should get you back to floating point basic but don't do it yet. The high res subroutines also overlap the floating point basic, so I had to make changes there and load it in the area hex 3C00 to 3FFF. See Listing 2 for those changes. Now that I made all the changes in monitor, 0G RETURN gets me back to floating point basic where I load the program as usual. The first time I did this I was surprised it worked, but it did. I hope the configuration of your system will allow you to use this program without moving things around, but if you do these changes are easier than it looks. Saving the new high res subroutines on tape or disk will speed things up the next time you have to do this.

If you're wondering how accurate this program is, I used an almanac for 1980 that gave the dates of special events for the planets, and all 20 dates that I tried worked. The display that I got for each date corresponded to what the almanac said was happening. I also have a book that gives the location of the planets 22 years ago, and the display I got was accurate enough not to make changes to the program.

This program can still be used on micros without high res graphics. The plotting routines can be changed to print the values RV, V1, and the X,Y positions. The distances between the Earth and other planets can also be calculated. If you have any questions or problems, don't hesitate to write, but please include a SASE or two stamps for postage and envelopes to guarantee a response. If there are a lot of questions, I don't want to go broke on postage.

One last word about the display. Using Figure 4 and the July 12th date, a line drawn form the Earth thru the planet Venus to the 10 degree circle intersects at about 47 degrees. This doesn't correspond to where Venus appears in the sky. Since the 10 degree circle doesn't have an infinite radius and is centered on the Sun and not the Earth, a line from the Sun parallel to the first should be drawn. This intersects the 10 degree circle at about 76 degrees. Looking a Figure 3, 76 degrees is in the constellation Taurus, and corresponds to where Venus should appear on July 12th.

μ

Dave Partyka works as a programmer on an IBM 3031 OS system for the May Department Store Company. He has been programming for three years, and he has been an operator for four years prior to that. Before he began work at the May Company he servde four years in the US Navy where he worked in data processing.

	Sidereal revolution in days	Distance from Sun in million miles max.	min.	Longitude of perhelion in degrees	Eccentricity
Mercury	87.969	43.403	28.597	77.1	.2056
Venus	224.701	67.726	66.813	131.3	.0068
Earth	365.256	94.555	91.445	102.6	.0167
Mars	686.980	154.936	128.471	335.7	.0934
Jupiter	4332.125	507.046	460.595	13.6	.0478
Saturn	10825.863	937.541	838.425	95.5	.0555
Uranus	30676.15	1859.748	1699.331	172.9	.0503
Neptune	59911.13	2821.686	2760.386	58.5	.0066
Pluto	90824.2	4551.386	2756.427	223.0	.2548

Listing 1.

ADDRESS	OLD	NEW
C01	20	40
C65	0B	3B
C7E	0C	3C
CE3	0D	3D
D0A	0D	3D
D62	0D	3D
D6B	0D	3D
D93	0D	3D
D9F	0D	3D
DCD	0E	3E
DD5	0E	3E
DF6	0D	3D
E02	0D	3D
E3D	0D	3D
EBF	0C	3C
EC6	0E	3E
EC9	0C	3C
ED8	0C	3C
EF1	0D	3D

From monitor load the high res subroutines in the normal location, C00 to FFF. Make these changes then move the subroutines to 3C00 by keying 3C00<C00.FFFM then RETURN.

The value in location C01 was changed to use page 2 (4000-6000) instead of page 1 (2000-4000).

OLD VALUES DEC.	HEX.	HIGH-RES COMMANDS	NEW VALUES DEC.	HEX.
3072	C00	INIT	15360	3C00
3086	C0E	CLEAR	15374	3C0E
3780	EC4	PLOT	16068	3EC4
3761	EB1	POSN	16049	3EB1
3786	ECA	LINE	16074	3ECA
3805	EDD	SHAPE	16093	3EDD

Listing 2.

```
1    REM SOLAR SYSTEM SIMULATION FOR THE APPLE II
2    REM WRITTEN BY  DAVE A. PARTYKA
3    REM             1707 N. NANTUCKETT DR.
4    REM             LORAIN, OHIO  44053
5    REM WRITTEN FEB. 1980
10   GOTO 1000
90   REM (100-110) POKE X AND Y VALUES FOR PLOTTING
100  POKE 800,X-INT(X/256)*256:POKE 801,INT(X/256)
110  POKE 802,Y:L=USR(16068):RETURN
150  REM (200-300) CALCULATE THE X AND Y PLANET POSITIONS
200  D=Z-INT(Z/SRD)*SRD
205  REM D IS FOR DAYS
210  B=Q-(D/SRD*Q2)
220  RV=A-(P/(1+E*COS(B)))
225  REM RV IS THE RADIUS VECTOR OR DISTANCE FROM
         THE SUN TO THE PLANET
230  V=PE/RV-EZ
240  IF V=>1 THEN V=VL
245  IF V=<-1 THEN V=-VL
250  V1 =-ATN(V/SQR(-V*V+1))+T
255  REM V1 IS THE ANGLE THAT THE PLANET
     LIES FROM THE SUN. THE 0 POINT BEING AT
     THE RIGHT, INCREASING COUNTER CLOCKWISE.
260  IF D SRD/2 THEN V1=Q2-V1
```

```
270  V1=V1+J
280  X=COS(V1)*RV:Y=-SIN(V1)*RV*FA
290  X=X*TT+X1:Y=Y*TT+Y1
300  RETURN
900  REM (1000) DISPLAY PRIMARY PAGE, SET TEXT MODE
1000 POKE-16300,0:POKE-16303,0
1010 T=1.5708
1020 Q=3.14159265
1030 Q2=6.2831853
1040 VL=.99999999
1050 FA=29/32
1055 REM FA IS THE RATIO OF X TO Y TO PLOT A CIRCLE
     ON THE APPLE INSTEAD OF AN OVAL
1060 X1=140:Y1=96
1700 PRINT:PRINT:PRINT:PRINT:PRINT
1800 PRINT "DO YOU WANT TO DISPLAY "
1810 PRINT:PRINT "THE SAME PLANETS AS YOUR LAST RUN"
1815 PRINT:INPUT "Y OR N ";A$
1820 PRINT:PRINT
1830 IF A$="N" THEN 2000
1840 IF A$<>"Y" THEN 1800
1850 IF S1<>0 THEN 4000
1855 PRINT:PRINT
1860 PRINT:PRINT "YOU HAV'NT PICKED THE PLANETS YET"
1870 PRINT:PRINT:PRINT
2000 PRINT "CHOOSE THE PLANETS YOU WANT TO DISPLAY"
2005 PRINT
2010 PRINT "ENTER A 1 FOR YES, 0 FOR NO"
2011 PRINT
2012 REM (2020-2079) GET SPECIFIC VALUES FOR EACH PLANET
2013 REM S1=ORBITAL PERIOD: P1=A1*(1-E1*E1)/2
2014 REM E1=ECCENTRICITY: U1=P1/E1: K1=1/E1
2015 REM A1=MINIMUM + MAXIMUM DISTANCE FROM SUN
2016 REM J1=LONGITUDE OF PERIHELION IN RADIANS
2017 REM W1=DAYS FROM 0 DEGREES TO PERIHELION FOR 1980
2018 REM TT=SCALING FACTOR TO USE FULL PLOTTING AREA
     IF SELECTED PLANETS ARE DISPLAYED.
2020 INPUT "DISPLAY MERCURY      ";ME
2021 S1=87.969
2022 E1=.2056
2023 A1=43.403+28.597
2024 P1=A1*(1-E1*E1)/2
2025 K1=1/E1
2026 U1=P1/E1
2027 J1=77.1*Q/180
2028 W1=37.58
2029 IF ME=1 THEN TT=2.3
2030 INPUT "DISPLAY VENUS        ";VE
2031 S2=224.701
2032 E2=.0068
2033 A2=67.726+66.813
2034 P2=A2*(1-E2*E2)/2
2035 K2=1/E2
2036 U2=P2/E2
2037 J2=131.3*Q/180
2038 W2=140.5
2039 IF VE=1 THEN TT=1.5
2040 INPUT "DISPLAY EARTH        ";EA
2041 S3=365.256
2042 E3=.0167
2043 A3=94.555+91.445
2044 P3=A3*(1-E3*E3)/2
2045 K3=1/E3
2046 U3=P3/E3
2047 J3=102.6*Q/180
2048 W3=-3
2049 IF EA=1 THEN TT=1.05
2050 INPUT "DISPLAY MARS         ";MA
2051 S4=686.980
2052 E4=.0934
2053 A4=154.936+128.471
2054 P4=A4*(1-E4*E4)/2
2055 K4=1/E4
2056 U4=P4/E4
```

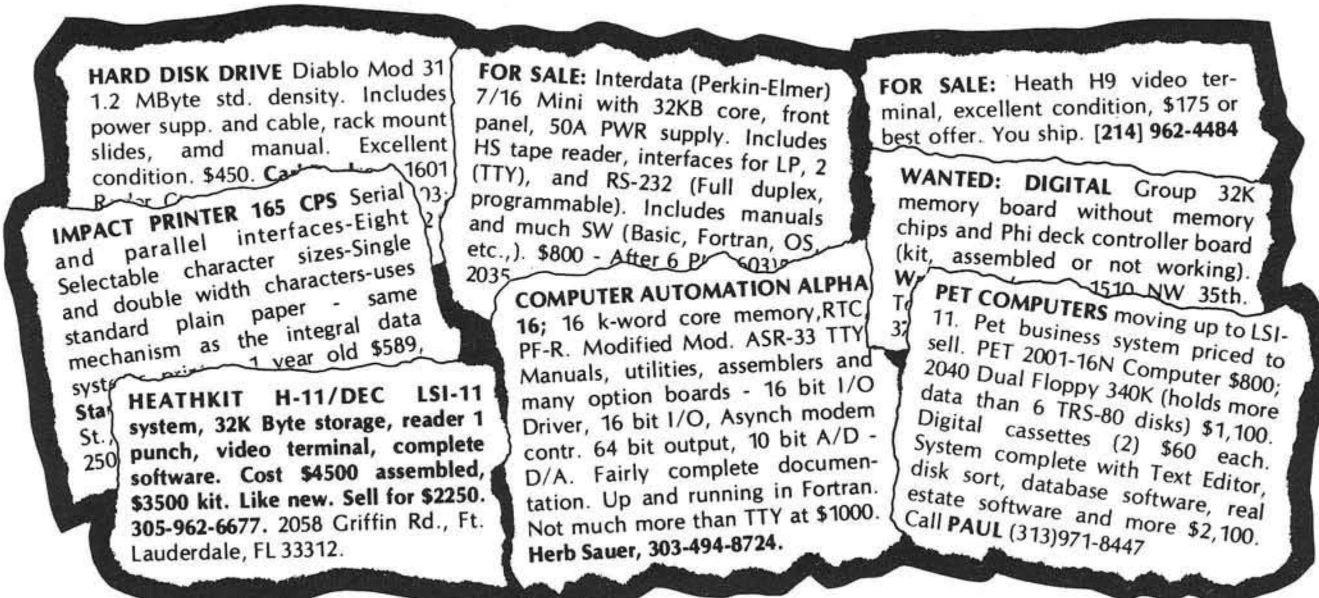
```
2057 J4=335.7*Q/180
2058 W4=289
2059 IF MA=1 THEN TT=.6
2060 INPUT "DISPLAY JUPITER      ";JU
2061 S5=4332.125
2062 E5=.0478
2063 A5=507.046+460.595
2064 P5=A5*(1-E5*E5)/2
2065 K5=1/E5
2066 U5=P5/E5
2067 J5=13.6*Q/180
2068 W5=1604
2069 IF JU=1 THEN TT=.19
2070 INPUT "DISPLAY SATURN       ";SA
2071 S6=10825.863
2072 E6=.0555
2073 A6=937.541+838.425
2074 P6=A6*(1-E6*E6)/2
2075 K6=1/E6
2076 U6=P6/E6
2077 J6=95.5*Q/180
2078 W6=2115
2079 IF SA=1 then TT=.1
3900 PRINT:PRINT:PRINT
4000 PRINT:PRINT "DO YOU WANT":PRINT
4010 INPUT "POINT (0) OR CONTINUOUS (1) PLOTS ";TY
4015 IF TY <> 0 AND TY <> 1 THEN 4000
4020 PRINT:PRINT:PRINT
4030 PRINT:PRINT "DO YOU WANT TO START AT":PRINT
4040 PRINT "A SPECIFIC DATE (0) ":PRINT
4050 INPUT "OR THE BEGINNING OF THE YEAR (1) ";DT
4051 IF DT<>0 AND DT<>1 THEN 4020
4052 IF DT=1 THEN 4060
4053 PRINT:PRINT:PRINT
4054 INPUT "ENTER # OF DAYS SINCE JAN 0 1980 ";DE
4057 Z1=DE
4060 PRINT:PRINT:INPUT "ENTER # OF DAYS TO PLOT ";DN
4070 PRINT:PRINT:PRINT
4080 INPUT "ENTER # OF DAYS BETWEEN PLOTS ";DA
4082 IF DA<>0 THEN 4800
4084 PRINT:PRINT
4086 PRINT "0 NOT ALLOWED:GOTO 4070
4090 REM (4800) INIT HIGH RES, FULL SCREEN, PAGE 2
4800 L=USR(15360):POKE-16302,0:POKE-16299,0
4802 REM (4805-4860) PLOT REFERENCE POINTS AND OUTER
     10 DEGREE CIRCLE
4805 POKE 812,255
4810 X=140:Y=96:GOSUB 100
4811 X=141:Y=96:GOSUB 100
4815 X=248:Y=96:GOSUB 100
4820 FOR L1=0TOQ2STEP1/36*Q2
4830 X=X1+COS(L1)*105.9
4840 Y=Y1-SIN(L1)*105.9*FA
4850 GOSUB 100
4860 NEXT L1
4900 REM (5100-5140) SET UP VALUES FOR MERCURY AND PLOT
5100 IF ME=0 THEN 5200
5110 A=A1:P=P1:E=E1:PE=U1:EZ=K1:SRD=S1:J=J1:W=W1:Z=Z1+W
5120 GOSUB 200:F1=X:G1=Y
5125 IF TY=1 THEN 5140
5130 X=M1:Y=N1:POKE812,0:GOSUB 100
5140 X=F1:Y=G1:M1=X:N1=Y:POKE812,255:GOSUB 100
5190 REM (5200-5240) SET UP VALUES FOR VENUS AND PLOT
5200 IF VE=0 THEN 5300
5210 A=A2:P=P2:E=E2:PE=U2:EZ=K2:SRD=S2:J=J2:W=W2:Z=Z1+W
5220 GOSUB 200:F2=X:G2=Y
5225 IF TY=1 THEN 5240
5230 X=M2:Y=N2:POKE812,0:GOSUB 100
5240 X=F2:Y=G2:M2=X:N2=Y:POKE812,255:GOSUB 100
```

```
5290 REM (5300-5340) SET UP VALUES FOR EARTH AND PLOT
5300 IF EA=0 THEN 5400
5310 A=A3:P=P3:E=E3:PE=U3:EZ=K3:SRD=S3:J=J3:W=W3:Z=Z1+W
5320 GOSUB 200:F3=X:G3=Y
5325 IF TY=1 THEN 5340
5330 X=M3:Y=N3:POKE812,0:GOSUB 100
5340 X=F3:Y=G3:M3=X:N3=Y:POKE812,255:GOSUB 100
5390 REM (5400-5440) SET UP VALUES FOR MARS AND PLOT
5400 IF MA=0 THEN 5500
5410 A=A4:P=P4:E=E4:PE=U4:EZ=K4:SRD=S4:J=J4:W=W4:Z=Z1+W
5420 GOSUB 200:F4=X:G4=Y
5425 IF TY=1 THEN 5440
5430 X=M4:Y=N4:POKE812,0:GOSUB 100
5440 X=F4:Y=G4:M4=X:N4=Y:POKE812,255:GOSUB 100
5490 REM (5500-5540) SET UP VALUES FOR JUPITER AND PLOT
5500 IF JU=0 THEN 5600
5510 A=A5:P=P5:E=E5:PE=U5:EZ=K5:SRD=S5:J=J5:W=W5:Z=Z1+W
5520 GOSUB 200:F5=X:G5=Y
5525 IF TY=1 THEN 5540
5530 X=M5:Y=N5:POKE812,0:GOSUB 100
5540 X=F5:Y=G5:M5=X:N5=Y:POKE812,255:GOSUB 100
5590 REM (5600-5640) SET UP VALUES FOR SATURN AND PLOT
5600 IF SA=0 THEN 6000
5610 A=A6:P=P6:E=E6:PE=U6:EZ=K6:SRD=S6:J=J6:W=W6:Z=Z1+W
5620 GOSUB 200:F6=X:G6=Y
5625 IF TY=1 THEN 5640
5630 X=M6:Y=N6:POKE812,0:GOSUB 100
5640 X=F6:Y=G6:M6=X:N6=Y:POKE812,255:GOSUB 100
6000 Z1=Z1+DA
6100 IF Z1>DE+DN THEN 7000
6200 GOTO 5100
7000 X=279:Y=190:GOSUB 100:INPUTA$
7050 REM (7000) PLOT POINT 297 190 TO INDICATE END OF
     SIMULATION THEN WAIT FOR INPUT OF ANY CHARACTER
     TO START AGAIN.
7100 Z1=0:DE=0
7200 GOTO 1000
```

Attention

BARGAIN HUNTERS

Receive Hundreds of Classified Ads Like These Every Month



COMPUTER SHOPPER, the new buy, sell, and trade publication, is ready to help you with the latest information on personal, small business and large-system computers, accessories and software.

Each ad-packed issue is full of bargains you are looking for. Included are ads from individuals throughout the United States who are selling their good, pre-owned equipment just so they can trade-up to new equipment coming on the market.

But, COMPUTER SHOPPER'S bargains won't be yours unless you subscribe. This useful, money-saving publication can become your way to communicate with other buyers, sellers, and traders all over the nation.

Whether you are a hobbyist or a part-time user, COMPUTER SHOPPER will put you in touch with the nationwide computer marketplace in time for you to take advantage of bargain opportunities.

Have something to sell? A COMPUTER SHOPPER subscriber probably wants to buy it.

Looking for a part, component or even a complete system? A COMPUTER SHOPPER subscriber probably wants to sell it.

COMPUTER SHOPPER is THE marketplace for anything in computers and is read by thousands of people who are ready to buy.

COMPUTER SHOPPER offers a unique format in which classified ads are categorized for fast location of specific items. Combining this with low individual ad rates — 10 cents a word — makes it the ideal place for buyers and sellers to communicate. And, its mix of individual, dealer, and manufacturer ads enable subscribers to find what they want at the best price possible.

COMPUTER SHOPPER will work for you in other ways, too. If you are just thinking about getting into computers, it can help you learn product availability and prices before you make a decision. And, through the timely ads, COMPUTER SHOPPER will keep you abreast of changes in the market which could create bargain opportunities for you.

BUT COMPUTER SHOPPER cannot work for you unless you subscribe.

Want to look us over first? We'll give you your first issue FREE and then bill you for the next 12. If you are not convinced COMPUTER SHOPPER suits your needs, just write "cancel" on the invoice and return it.

And, to let COMPUTER SHOPPER start working for you right now, with a paid subscription we'll also give you a FREE classified ad to sell your pre-owned equipment or to find equipment you want

If you don't need to use the free classified ad now, use it anytime during your subscription.

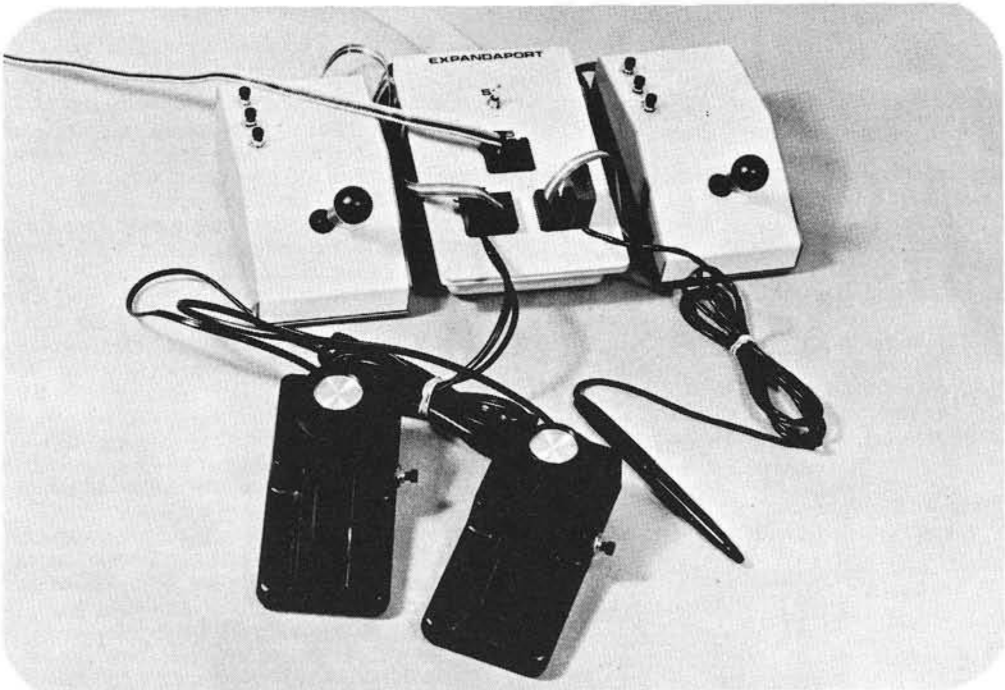
Subscription: $10/year, 12 issues plus your first free one. Bank cards accepted. Money back guarantee.

The Nationwide Marketplace for Computer Equipment

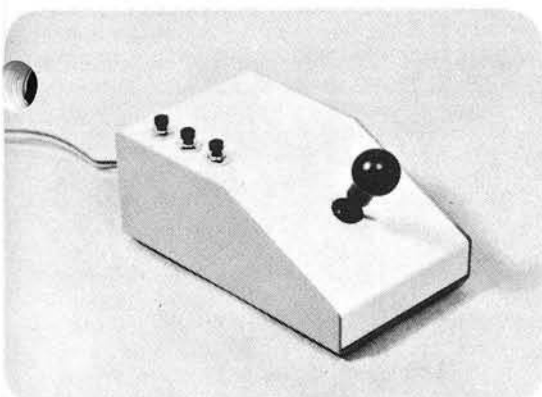
COMPUTER SHOPPER

P.O. Box F22 • Titusville, Florida 32780
Telephone 305-269-3211

JOYSTICK & EXPANDA-PORT FOR YOUR APPLE II® COMPUTER

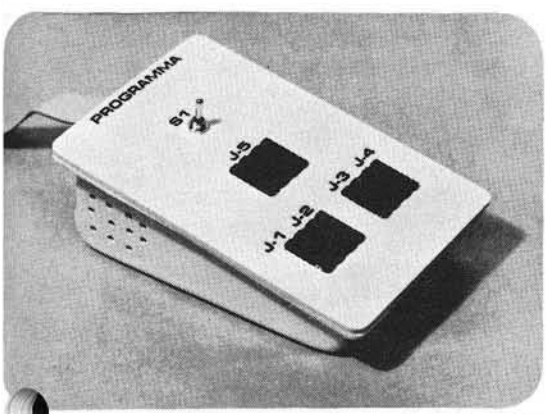


EVERY APPLE II® OWNER SHOULD HAVE ONE!



JOYSTICK $49.95

The PROGRAMMA JOYSTICK is an input peripheral that attaches to the APPLE II Computer's game I/O Port. The JOYSTICK is a must for the serious game player, and it offers a degree of linearity not currently available with other joysticks. The ease of manueverability and the availability of the "functional" switches make the PROGRAMMA JOYSTICK a much needed enhancement to any APPLE II Computer System owner. The PROGRAMMA JOYSTICK comes completely assembled and tested.



EXPANDA-PORT $59.95

The PROGRAMMA EXPANDA-PORT is a multi-port expander for the game I/O port of any APPLE II Computer System. In addition to allowing expansion for up to six devices, the EXPANDA-PORT contains a built-in speaker that replaces the function of the Apple II's speaker. The switch on the EXPANDA-PORT allow for the selection of the specific device desired and for the switching of that device. No unplugging of any device connected to the EXPANDA-PORT is required. The PROGRAMMA EXPANDA PORT comes completely assembled and tested.

The PROGRAMMA JOYSTICK and EXPANDA-PORT are available on a limited basis through your local computer dealer. Apple II is a registered trademark of Apple Computers, Inc.

PROGRAMMA INTERNATIONAL, INC.
3400 Wilshire Blvd.
Los Angeles, CA 90010 (213) 384-0579 · 384-1116 · 384-1117

PROGRAMMA

Computer System Products

In this, the third issue of the Ohio Scientific Small Systems Journal in MICRO, we are happy to present the first user contributed article the Journal has ever featured.

The first article, based on a contribution by Phil Lindquist of Union Lake, Michigan, features a short but high performance word processor program and some insight into its operation. This high utility program may find use in your program library. We shall be happy to feature other contributions of this quality in future issues.

The second article is on hardware ROM (Read Only Memory) configurations used in Ohio Scientific Systems. The article may be somewhat detailed in technical aspects of the hardware, but if you need to know this information, there is no better way than getting into the details here.

Reader suggestions on article content are welcome. Please submit suggestions or other contributions to:

OHIO SCIENTIFIC, INC.
Small Systems Journal
1333 S. Chillicothe Rd.
Aurora, Ohio 44202

MINIMICRO WORD PROCESSOR

A simple word processor with great utility has been written by Phil Lindquist of Union Lake, Michigan. The program provides:
(a) Resequencing of text lines,
(b) Right and left hand justification of text, which gives the crisp look of aligned margins,
(c) centering of titles and subheadings, and
(d)by passing the text editor for selected lines.

For many text editing tasks, high quality printed output can be obtained with few program constraints. It's a good introduction to word processing.

INTIALIZATION

The Minimicro program is listed in Listing 1. To appreciate its features, enter the program as listed. It will run on all 8" floppy 65D V3.X systems. For casette based systems, changes should be made for screen printng.
C-1-P or C-4-P Cassette Systems: Change the following lines to:

```
11010 AD = 121:SN = 100:INC = 10
12040 REM DEVICE IS SPECIFIED HERE
12160 AD = 121
12500 PRINT,TAB(LM); 0$: GOTO 12200
```

C-1-P or C-4-P MF Systems: Change the following lines to:

```
11010 AD = 12921:SN = 100:INC = 10
12040 REM DEVICE IS SPECIFIED HERE
12160 AD = 12921
12500 PRINT,TAB(LM);0$:GOTO 12200
```

If you wished to use a line printer, the form of the original listing shows the method. Line 12500 is the only text output. With these minor changes, let's look at the operation of the program.

USE:

The text editor program is invoked by the BASIC command
RUN 10000
at which time you are given the choice of
RENUMBER/LIST/EXIT?
For any other choice, we'll need some text to test the program. Therefore, before you run the program, enter your text with sequential line numbers. Text will be indicated by a quote followng the line number, such as
10 "NOW IS THE TIME
To have a sequence of text renumbered, such as
10"NOW IS THE TIME
20"FOR ALL GOOD
30"MEN TO AID
we again type
RUN 10000
and respond to the text editor's selection list by
RENUMBER
The text is now resequenced, starting with statement 100, incrementing statement numbers in steps of 10. Program statement 11010 may be modified to permit other starting statement numbers (SN) than 100 and other incrementing steps (INC) than 10.

The final feature to be examined is the selection LIST in the text editor's selection list. Within LIST, four symbols are used to control the text editor. These symbols are

(a)the quotes("), following a statement number, indicate that thestatement is text

(b)the slash (/) character, immediately following the quotes, indicates that the line is to be centered within the allowed margins. An example of this feature would be
10"/TITLE

(c)the period (.) character, immediately following the quotes indicates that the line is to be listed without the use of text editor text alignment.

(d) the apostrophe (') character, when typed twice in succession, will be printed as the quote (") symbol. This convention was necessary because we have already used the quotation mark to delimit text.

Lets's type some text and see how the editor responds. First, some sample text might be
100"/TITLE
110"NOW IS THE TIME
120"FOR SEVERAL GOOD
130"PERSONS WHO
140".DO NOT EDIT THIS
Request the editor's service by typing
RUN 1000
and respond with
LIST
it will request input with
FIRST LINE, LAST LINE?
Respond with
100,140
(Note: 0,9999 will LIST all text)
The editor will then ask
LEFT MARGIN, RIGHT MARGIN?
Since all text lines were less than 20 characters long, I chose a left margin of 10 spaces and the right margin 20 spaces later, as
10,30
The resulting text appears as

```
                    TITLE
NOW IS THE TIME
FOR SEVERAL GOOD
```

PERSONS WHO
DO NOT EDIT THIS

The title is centered midway between columns 10 and 30. The next two lines have spaces added to pad them to fit exactly between the column 10 and column 30 margins. Text line 130 is quite short, so the text editor does not adjust the text margins. Finally, the last line, 140, will not be edited, because the leading period caused the editor to ignore the statement.

Program Method:

Let's take a look at the method which was used in this program to align margins. The line width, right margin minus the left margin, is computed in line 12150. If the present line is greater in length than the computed line width (between margin spacing), (line 12420), then the line is printed "as is". Only in the cases for text line length in lines 12430 to 12480, is the text line is padded out with blanks. Because the line is justified going from right to left, we don't add padding on top of padding, giving uneven spacing. This scheme distributes the spaces evenly across the line, in between words. Using repeated passes through the program to add one space between words, until margins are aligned, gives even spacing.

In the case of titles to be centered, statement 12490 subtracts the character string length to be printed from the inter-margin width. Since this is the number of blank spaces needed, a character string of half this length, filled with blanks is placed in front of the title to be printed.

Some useful information can be gained by looking at the method used to address text. The text is stored sequentially as a part of the Minimicro program. The first two locations (bytes) at the start of each line of text contain the address of the next line of text. In the Minimicro program, the variable AD starts off pointing to the start of the Basic program, and therefore to the text to be edited. Each time we examine a new line of text, the value of AD is updated by the latest value of the program pointer to the text. When we reach the last line of text, two bytes, of value 00, are found in the first two bytes of that line of text (placed there by the BASIC editor); this will label the line as the last line of text. Each line of text has the end of the text line delimited by a 0 (not the ASCII symbol for a 0). The use of the end of text line delimiter and the end of text delimiter permit easy tests for word processing.

With these insights into a very useful program contributed by Phil Lindquist, you may see ways to incorporate this program into your programs or library. Thanks Phil!

Listing 1

```
10000 PRINT ''Minimicro Word Processor, Number 2''
10010 PRINT ''Program by P. Lindquist, May 1980''
10020 PRINT
10030 PRINT
10040 INPUT ''RENUMBER/LIST/EXIT'';A$: PRINT
10050 IF LEFT$(A$,1)=''R'' THEN 11000
10060 IF LEFT$(A$,1)=''L'' THEN 12000
10070 IF LEFT$(A$,1)=''E'' THEN STOP
10080 GOTO 10020
11000 REM *** RESEQUENCE
11010 AD=12665: SN=100: INC=10
11020 AD=PEEK(AD)+256*PEEK(AD+1)
11030 LN=PEEK(AD+2)+256*PEEK(AD+3)
11040 IF LN > 9999 THEN 10020
11050 BT=INT(SN/256): POKE AD+3,BT
11060 BT=SN-256*BT: POKE AD+2,BT
11070 SN=SN+INC
11080 GOTO 11020
12000 REM *** JUSTIFIED LIST
12010 PRINT ''LINE NUMBER RANGE''
12020 INPUT ''FIRST LINE, LAST LINE'';FL,LL
12030 PRINT
12040 INPUT ''DEVICE NUMBER FOR OUTPUT'';DV
12100 REM
12110 REM USE THIS AREA FOR SPECIAL OUTPUT INIT IF REQUIRED
12120 PRINT
12130 INPUT ''LEFT MARGIN, RIGHT MARGIN'';LM, RM
12140 PRINT
12150 RM=RM-LM: IF RM < 1 THEN 10020
12160 AD=12665
12200 AD=PEEK(AD)+256*PEEK(AD+1)
12210 IF AD=0 THEN 10020
12220 BP=AD+4
12230 0$=''''
12240 LN=PEEK(AD+2)+256*PEEK(AD+3)
12250 IF LN > LL THEN 10020
12260 IF LN < FL THEN 12200
12270 IF PEEK(BP)=34 THEN BP=BP+1
12300 CH=PEEK(BP)
12310 IF CH=0 THEN 12380
12320 0$=0$+CHR$(CH)
12330 LN=LEN(0$)
12340 IF LN < 2 THEN 12360
12350 IF RIGHT$(0$,2)=''"''THEN 0$=LEFT$(0$,LN-2)+CHR$(34)
12360 BP=BP+1
12370 GOTO 12300
12380 LN=LEN(0$)
12390 IFLEFT$(0$,1)=''/''THEN0$=RIGHT$(0$,LN-1):GOTO12490
12400 IFLEFT$(0$,1)=''.''THEN 0$=RIGHT$(0$,LN-1):GOTO12500
12410 IF 10*LN< 7*RM THEN 12500
12420 IF LN > =RM THEN 12500
12430 FOR I=LN TO 1 STEP -1
12440 IF MID$(0$, I,) < > '' '' THEN 12480
12450 0$=LEFT$(0$,I)+'' ''+RIGHT$(0$,LN-I)
12460 LN=LN+1
12470 IF LN > =RM THEN 12500
12480 NEXT : GOTO 12430
12490 LN=INT((RM-LN)/2): FOR I=1 TO LN: 0$='' ''+0$: NEXT
12500 PRINT #DV,TAB(LM);0$: GOTO 12200
```

Ohio Scientific System ROMS

Most users of Ohio Scientific computers are aware that the C1P, C4P, and C8P systems all contain 8K BASIC in ROM (Read Only Memory). What many users are unaware of is that, in addition to the BASIC ROMs, there is also a separate ROM used for the system monitor code. Additionally, all floppy and hard disk based systems also contain a system and "boot" ROM. The boot code in ROM is used to bring up the system from disk.

This article will cover two main topics. First, an overview of the general ROM decoding scheme will be presented. In the second section, the content and use of the four currently available Ohio Scientific system ROMs will be discussed.

ROM Decoding

The system ROMs used in Ohio Scientific computers are all 16384 bit parts. They are normally referred to as

Interface of OSI-C1P With Heath Printer

This article provides all of the hardware and software information that is required to implement and utilize this combination.

William L. Taylor
246 Flora Rd.
Leavittsburg, OH 44430

Most personal computer users want to attach peripherals to their machines. These peripherals take the form of disk drives, cassette tape systems, printers, etc. The cassette drive, the disk and the printer usually have the first priority over other peripherals. Most personal computers have a cassette systems as part of the I/O in the system, so a printer or disk drive is the user's next choice.

In my case the first choice peripheral was a printer. This was needed to aid me in article writing and to provide quality program hard copy listings for these articles. I researched the features of the available printers on the market for cost effective comparisons, and decided to purchase the Heath H14 for use with my Ohio Scientific C1P.

The purpose of this article will be to give the reader and the user of the OSI C1P the needed information for hardware and software implementation to allow the C1P and H14 to operate as a system. For this, there are some hardware additions that must be included on the C1P's circuit board and one modification to the H14 printer. These additions and modifications will be explained in this article.

The first of these steps will be the RS232C addition that is needed for the C1P. The modifications to follow have been included in the author's system and the proof is in the listings of the programs in this article and other articles that have appeared in Micro. But I must remind the reader that when one does the modifications to the C1P and printer, the warranties could be broken.

RS232 Implementation on the C1P

The Challenger C1P model 600 board can be configured for an RS232 output only for a printer. This port does not come with the components installed on the circuit board; the user must install them or have a dealer do it for him. The component count for this implementation is rather small in number, and is easily done. To have an RS232 printer port, you need only four resistors, one transistor, and a 25 pin female plug for the printer cable male connector. Also a 12 pin Molex connector to mate with the 12 pin Molex connector on the C1P's circuit board. These components can be easily purchased from most local electronics parts stores, or your local Radio Shack parts counter.

To begin the parts placement, be sure to remove the plug from the wall outlet. Next, remove the cabinet bottom. To do this, place the C1P with the top down on a soft surface. This will prevent damage to the surface of your C1P's cabinet. Remove the screws that hold the bottom in place, and remove the bottom cover by lifting it *straight* upward. This will expose the C1P' model 600 board and power supply. I found it best when doing this to remove the power supply leads from the model 600 board. If you do this, be sure to mark the points where the leads were soldered to the 600 board. Unsolder the the green, the black, brown and red leads from the circuit board. With these leads disconnected, the board can be completely removed from the cabinet for inspection and ease of installation of new components. To remove the 600 board, turn the C1P over to expose the keyboard. Remove the screws placed around the keyboard. (These are the only screws that hold the keyboard in place.)

With the keyboard placed on your workbench with the components and keyboard up, locate the page in your C1P' Users Manual that shows the component overlay diagram. This is the drawing that shows the complete board, and has the components drawn at their proper location. With the keyboard near you, the RS232 output port component location will be at the far end of the board. In the parts placement drawing, locate transistor Q1. Using the general circuit board trace drawing in Figure 1, and the schematic of Figure 1, install the extra components that make up your RS232 output port. These components are: Q1, R72, R63, R64, and R65. The IC,U62, is already installed on the board. There are no jumpers to add on the board, but one of the traces will have to be cut. This trace is marked W on the overlay drawing, and in the schematic of Figure 1. This trace is located near the end of R64 that goes to pin 7 of the circuit board connector J3. The trace W, is located on the foilside of the circuit board. Use a sharp knife such as an

Exacto to cut the trace. Be sure that the trace is completely cut and there is not any contact between the two ends. The next step in the modification of our C1P for the RS232 interface will be to add the 25 pin female D connector which will mate with the 25 pin male plug on the connecting cable from the printer. This connector will be mounted in the rectagular hole at the rear of the CIP's cabinet. But first, the connecting wires should be installed on the female connector and to a l2 pin Molex male connector. The l2 pin Molex connector will be used to mate the l2 pin female connector on the 600 circuit board at J3. Follow the schematic in Figure l, connect and solder a l2 inch stranded wire from pin l of the 25 pin female connector to pin l of the l2 pin Molex male connector. Similarly, connect another l2 inch stranded wire from pin l7 of the 25 pin connector to pin 7 of the l2 pin Molex connector. Finally, connect a l2 inch stranded wire from pin 3 of the 25 pin connector to pin 2 of the Molex connector. This completes the wiring of the plugs.

The next step is to install the 25 pin female D connector in the rectangular hole at the rear of the C1P'scabinet. You will need two 6/32 screws and nuts for this step.

Check your wiring against the schematic of figure 1, the parts overlay in the C1P User's Manual, and the parts placement trace drawing in figure 1 for correctness. Check closely the printed circuit board foil side for solder bridges. Be sure Q1 is at the correct location and placed properly on the board. That is, be sure the emitter, the base, and the collector leads are correctly soldered to the circuit board. Reconnect the power leads if they were disconnected. This completes the parts installation. The circuit board can now be re-installed in the C1P's cabinet with the screws previously removed.

Locate the position of J3, pin 1. Plug in the l2 pin Molex male connector into J3. Re-install the 6 pin Molex connector at J2. Be sure that the plugs are properly orientated before they are plugged into their respective socket. Re-solder the power leads if they were removed at the beginning of the modification. Replace the bottom cover on the C1P with the screws that were previously removed. This completes the parts installation on your C1P.

Modifying the H14 and Bringing Up the System...

As previously stated, the H14 Printer will require one simple modification. This consists of locating a wire, cutting it, and attaching a short wire to the minus 12 volt power supply in the H14 Printer. Locate the black/red wire going to pin 8 of the main circuit board connector for the I/O cable. Cut the black/red wire about one inch from the 15 pin female connector. This should be done so that the wire could be re-attached in the event that the printer is sold in the future.

Using a short piece of small stranded hookup wire, connect one end to the black/red wire going to the I/O cable. Solder the connection. At-

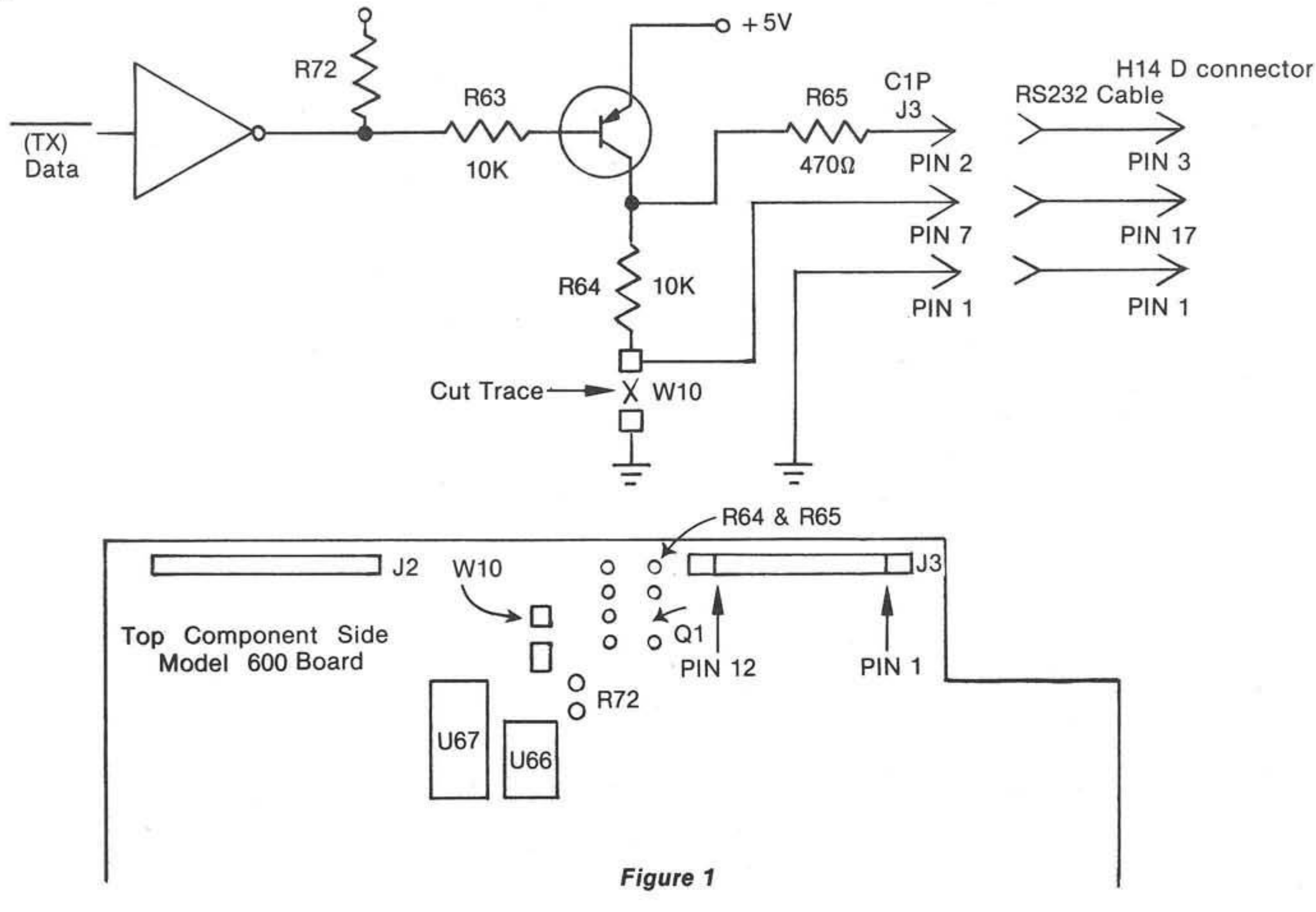


Figure 1

tach the other end of this wire to the negative end of capacitor C113 on the main circuit board in the H14 printer. This capacitor is located just to the rear of the high temperature pot, R195. This completes the modification to the printer. Next, set the baud rate switches for the RS232 interface operation. The jumpers are located at locations J114 to J115 and J110 to J109. This configures the H14 for RS232 input.

At this point you should have a working RS232 output port on your C1P, a plug that will mate the C1P and the plug from the H14 printer. Plug the printer cable into the female connector on the C1P. Connect your Monitor and cassette cables and place the power cables for the printer and the C1P into the wall outlet. Turn both units on. Reset the C1P. Bring up BASIC. Type in a BASIC program line such as, 10 PRINT " THIS IS A TEST OF THE C1P AND THE H14 PRINTER." Place the H14 ON-LINE. Place the C1P in the SAVE mode. When you hit a carriage return the H14 printer should respond with the word SAVE followed by OK. Next, type RUN followed by a carraige return. The H14 should respond with the message that was entered as your BASIC program line. If all went well, you now have working system using the Ohio Scientific Challenger C1P and the Heath H14 printer using the RS232 port on the C1P.

General System Description

The RS232 output port on the C1P services the H14 Printer in the same manner as the Cassette port services the cassette recorder. That is, when the user wishes to save a program on tape the key word SAVE followed by a carraige return and then the keyword LIST and carraige return. The program of interest will be listed out and written on tape. With the RS232 port and the H14 printer connected and On-Line the program will be written on paper for a hard copy record.

If the user wants either a hard copy or a tape, it is a simple method. Either use the cassette recorder for tapes or the printer for the hard copy. The H14 can be used in the RUN mode of a BASIC program. This is accomplished by placing the H14 On-Line and the C1P in the SAVE mode. When you run the BASIC pro-

```
f6C+
LIST
 1 REM BASIC HEX MEMORY DUMP FOR HEATH 14 LINE PRINTER
 2 REM BY W. L. TAYLOR 5/1/79
 3 GOSUB 1000
 4 GOSUB 2000
 5 POKE 11,43:POKE 12,15
 6 X=USR(X)
 7 POKE 11,26:POKE 12,15
 8 X=USR(X)
 10 GOSUB 2200
 14 PRINT" BASIC HEX MEMORY DUMP FOR HEATH H14 PRINTER"
 15 PRINT:PRINT:PRINT
 20 POKE 11,00:POKE 12,15
 25 REM SET START AND END ADDRESSES
 30 INPUT "START ADDRESS";S
 40 INPUT "END ADDRESS";E
 45 C=0
 47 POKE 3894,S
 50 S=S*256:E=E*256
 54 POKE 11,00:POKE 12,15
 55 X=USR(X)
 56 POKE 3894,0
 57 X=USR(X)
 58 POKE 11,26:POKE 12,15
 59 X=USR(X):X=USR(X)
 60 POKE 11,00:POKE 12,14
 61 X=USR(X)
 62 POKE 11,26:POKE 12,15
 63 X=USR(X):X=USR(X)
 64 FOR A=S TO E
 65 POKE 11,00:POKE 12,15
 68 REM GET HEX CHARACTER
 70 B=PEEK(A)
 75 REM LOAD HEX CHARACTER IN BUFFER
 80 POKE 3894,B
 85 REM PRINT HEX CHARACTER
 90 X=USR(X)
 95 REM DO SPACE
 100 POKE 11,37:POKE 12,15
 110 X=USR(X)
 120 C=C+1
 125 REM CHECK FOR 16 CHARACTERS
 127 IF C=16 THEN 150
 130 NEXT A
 140 END
 145 REM DO CARRAGE RETURN AND LINE FEED
 150 POKE 11,26:POKE 12,15
 151 C=0:X=USR(X)
 152 GOTO 130
 160 X=USR(X)
 170 C=0
 180 GOTO 130
 999 REM LOAD MACHINE CODE ROUTINE FOR HEX DUMP
 1000 FOR G=3840 TO 3892
 1010 READ F:POKE G,F
 1020 NEXT G
 1025 RETURN
 1030 DATA 173,54,15,72,74,74,74,74,32
 1040 DATA 12,15,104,41,15,9,48
 1050 DATA 201,58,144,2,105,6,32,80,14
 1060 DATA 96,169,13,32,80,14,169
 1070 DATA 10,32,80,14,96,169,32,32,80,14
 1080 DATA 96,216,173,6,234,169,159,141,5,234,96
 2000 REM LOAD ACIA OUCH ROUTINE AT 0E50
 2002 FOR K=3664 TO 3676
 2005 READ Y:POKE K,Y
 2007 NEXT K
```

```
2008 RETURN
2010 DATA 72,173,0,240,74,74
2020 DATA 144,249,104,141,1,240,96
2199 REM LOAD MACHINE CODE FOR 16 COLUMNS
2200 FOR P=3584 TO 3646
2210 READ L:POKE P,L
2220 NEXT P
2225 RETURN
2230 DATA 162,0,232,189,15,14,32,80,14
2240 DATA 224,48,208,245,96,234,234
2250 DATA 48,32,32,49,32,32,50,32,32,51
2260 DATA 32,32,52,32,32,53,32,32,54,32
2270 DATA 32,55,32,32,56,32,32,57,32,32
2280 DATA 65,32,32,66,32,32,67,32,32
2290 DATA 68,32,32,69,32,32,70,32
OK
```

gram on your C1P, the printer will respond as does your monitor. Anything that is printed out to the Monitor screen will be printed on the H14. This is only one form of program operation that can be performed with the H14 attached to the C1P. The second use of the system is with programs written in BASIC or machine code to service the printer. Included in this article, is a program written in BASIC and machine code that will allow the user to do one of these tasks.

Software For The System

The following program will give the user of the H14 and the C1P system one of the working tools that will allow some special printing that will be useful and will demonstrate the use of the printer. The program in Listing 1, will let the user of this system explore the machine code routines that are resident in the C1P and also can be used to list the Hex contents of any user programs that should be written into the C1P.

The program in Listing 1, titled "BASIC Hex Dump For The Heath H14 Printer" was written to be a useful utility program. This program will allow the user with the C1P and the H14 printer to dump any 256 byte block of memory out to the printer.

The program uses many Machine Language calls through the USR function of BASIC. The BASIC portion of the program is used primarily for housekeeping. When the program is running, the user must enter the starting and ending addresses of the block of memory that are desired for printout. This is done at lines 30 and 40. The user must enter page numbers, such as, 16 etc. These page numbers are multiplied by 256 to arrive at the decimal equivalent that is needed for the BASIC variables. This is done at line 50. After the page numbers have been entered along with a carriage return, the printer will respond with a carriage return along with a line feed and a dump of the memory block. A general list of the modules in the program will be given, but a detailed description will not be necessary because each module in the program is separated by REM statements. The user may analyze the program simply by studying each module separately. All the Machine Code routines are loaded into memory on initialization. The Machine Code for the routines are stored in DATA statements and are loaded into memory with the POKE function of BASIC. The Machine Code routines are stored at 0E00 hex, 0E50 hex, and at 0F00 hex. A dump of the object code for the routines is given in Listing 2. This is also an example run of the format that the Hex Dump Program will produce. A list of the modules for the program follows. These modules are:

Line 1000 Machine code load routine for main hex dump.

Line 2199 Machine code for column numbering.

Line 2000 Machine code load of ACIA OUCH routine.

Line 145 Begin carriage return and line feed.

Line 95 Do space between each Hex characters.

Line 125 Check for 16 Hex characters.

The remainder of the modules can be found in the main BASIC program. Remember to set memory size to 3580 when bringing up BASIC.

In conclusion, this article has given the reader the needed information to allow the C1P to operate with an RS232 output port that can be used with any printer that has this type of input acceptance. In this article the example printer used was the Heath H14. This was my choice for a printer; yours may be one of some other manufacture. Keep in mind that there are several printers that could be used with the RS232 output port that is in foil on the C1P. It would be of advantage to do the needed parts implementation on your C1P. This article has been writ-

```
0E00
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F t

      A2 00 E8 BD 0F 0E 20 50 0E E0 30 D0 F5 60 EA EA
      30 20 20 31 20 20 32 20 20 33 20 20 34 20 20 35
      20 20 36 20 20 37 20 20 38 20 20 39 20 20 41 20
      20 42 20 20 43 20 20 44 20 20 45 20 20 46 20 F4
      EB CF D8 FB 5A E8 CE DA CB 48 DB 42 CF D3 FB E8
      48 AD 00 F0 4A 4A 90 F9 68 8D 01 F0 60 DF F5 B4
      85 94 E5 92 94 84 F4 BC E7 96 5C 33 96 9D F6 D0
      D8 98 90 C0 9C F4 C0 D8 F0 F4 D8 F4 4A 90 F8 C0
      FF 75 F0 F0 B0 7B FD 75 F4 E5 95 76 DF 54 D2 00
      FD 7D 73 FB F3 53 73 D3 EB 7D 70 F7 5B 51 F3 F1
      FD D5 78 7A DB DE D3 FA FA 1B F2 52 5F 13 7A D3
      F6 DD 77 77 76 D1 D7 35 D7 CE D6 F7 F1 40 33 B9
      9D 98 78 18. F5 F6 D3 DA 58 F8 91 FF FD 55 5C 88
      D9 5C CA C1 CB 4A DA B8 4B C2 EB D3 59 D9 DB FA
      1B 7D D8 78 FC 4A C2 DB D9 49 E8 DB D9 E9 F8 CC
      CD E5 D6 86 92 B7 99 DE D7 DC 99 D4 D6 87 F5 D1
```

ten to help simplify this task. Also, with the software provided in this article, to be specific, the hex dump program will work with any printer that can be used with the C1P. I hope that you can use this information and have learned with me.

Parts List C1P RS232 Port

1] Q1 PNP Transistor Radio Shack 276-2023
1] TRW 'JONES' Min. D Female Connector Number DB 25S
1] Resistor 1K Ohm 1/4 Watt
2] Resistor 10K Ohm 1/4 Watt
1] Resistor 470 Ohm 1/4 Watt
1] Connector Molex Male Number KKI56

Misc. 6/32 x 1/4 Screws
Wire, solder, etc.

μ

```
AD
0F00

0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F t

AD 36 0F 48 4A 4A 4A 4A 20 0C 0F 68 29 0F 09 30
C9 3A 90 02 69 06 20 50 0E 60 A9 0D 20 50 0E A9
0A 20 50 0E 60 A9 20 20 50 0E 60 D8 AD 06 EA A9
9F 8D 05 EA 60 34 34 D6 E6 D0 D6 F6 D5 27 F4 76
CD 79 F8 FB FA 5A 9B C9 1D 58 DF FB FB DB F9 C8
1C F7 86 A4 90 9F C4 DA 87 C3 9C 92 94 F5 BE D1
DB 6B D9 D9 BB FA 7A D0 C9 8B D5 D2 F3 BB D8 CA
F5 D5 C7 CC D6 C0 E2 D0 E1 95 87 F6 F5 93 FD F4
F7 F7 96 6E EC DE D0 F7 D2 99 8D D3 95 D3 DF BD
F3 70 DB C9 53 74 7B 52 D3 59 D3 FF E0 55 F1 69
77 F5 BD 9E B4 D6 B5 7E F0 F0 B4 9F 73 E5 F0 94
D7 8E C4 58 E0 D3 D7 44 F0 D8 54 7F 75 F5 61 E4
F7 F8 B4 90 F0 F8 B7 D6 A4 93 C6 56 A7 C6 B5 DC
59 4A 5B E8 DA 99 D9 E8 E9 0B 8C 71 C9 C5 DB 48
DD D8 5C 8C A1 F0 A8 DD E0 F2 1B D4 5C F4 71 F8
84 90 CA 00 8D 0A C9 CC 80 88 DF 00 58 AA 40 C8
```

Mr. Taylor has been using the OSI system computer since 1976 when he built his first system using OSI bare boards. This system consisted of the OSI 400 CPU, 480 Backplane board and the 440 video display board, along with an ASCII keyboard. With this system he learned to program in Machine Language.

He is interested in hardware for the C1P and software development. He is continually expanding the capabilities of the C1P, and most recently, he has interfaced the Heath H14 printer and added an interface to the OSI 48 line buss of his own design.

Applesoft Floating Point Routines

A discussion of where these important routines are located, what they do, and an example of their direct use.

R.M. Mottola
Cyborg Corp.
342 Western Ave.
Boston, MA 02135

Part of a recent project required me to write a routine that would calculate various statistical data reductions on a series of data points. The initial result, written in Applesoft floating point BASIC, worked well enough but took a healthy amount of time to execute. Upon doing some timing experiments, it became apparent that a good deal of the time required to perform the task was eaten up by BASIC overhead-conversion of types, floating point "FOR-NEXT" loops, and general interperter related functions.

What I really wanted was to write all of the routine in machine language. To do this, there were two options available. The first was to write some floating point routines which maintained the Applesoft five byte variable format. This proved to be impracical due to the amount of memory required for these routines.

The second and much more memory efficient solution was to locate the floating point routines already in my machine in Applesoft. This proved to be reasonably difficult for a number of reasons but after much head-scratching I've managed to unearth the following routines. Before using them, its probably a good idea to familiarize yourself with the format of both the Applesoft variables and the Applesoft floating point accumulators.

The format of Applesoft variables is a standard five byte floating point representation, with the highest order byte containing the exponent and the lower four bytes containing a signed mantissa. See page 137 of the Applesoft manual for more on this. The format of the Applesoft accumulators is a little different. You will notice from various Applesoft zero page usage tables that seven bytes have been allocated for each of the two floating point accumulators. The format of these accumulators is as follows: The highest order byte contains the exponent. The next four bytes contain the negative absolute value of the mantissa, as represented in Applesoft variable format. The sixth byte contains the origional high-order byte of the mantissa if a value has just been converted from variable format to accumulator format. In any case, this byte is used to represent the sign of the mantissa. The seventh and last byte of the accumulator is a "function" byte used in arithmetic operations. It is not initially assigned a value on conversion of a value from variable format-to a accumulator format.

To use the following floating point routines is a reasonably straight-forward process. For the sake of simplicity, you may find it easier to forget the accumulator formatting of values, and load all values into the accumulator using the "FPLOAD" subroutine listed. This routine performs the conversion while diong the load. You should also be careful to represent all values in normalized form. If you plan to use only values that have been previously specified by Applesoft, you will not have to do this as Applesoft normalizes all variables as they are specified. To use your own values, you may find the accompanying utility program useful.

Another thing to be careful about is floating point errors (Division by zero, Overflow). Since these floating point routines were not meant to be used outside of Applesoft, the entry points to the error handling routines are in ROM. Unfortunately, the vectors to these routines are cast in stone (or Silicone, anyway) and cannot be changed. There are two ways to deal with these errors:

1. Test your routines for "worst case" operation. If you can make sure that errors will never ocur, you've got it made.
2. Applesoft has the ability to vec-

tor errors to a specified Basic line number with the ONERR... GOTO STATEMENT TO DIRECT ERRORS TO A specified line number. On this line number, you can make a call to your own machine language error handling routines.

The following routines constitute the major arithmetic routines available in Applesoft. There are, of course, other fuctions buried in Basic which have not been identified here. I would appreciate hearing from anyone else who has dug into those mysterious ROMs.

Name: FPLOAD
Address: $EAF9
Symbolic: M⟶FPAC1

Loads variable into primary floating point accumulator. Converts to FPAC format. A and Y registers must point at variable in memory (ADL, ADH). Clears $AC.

Name: FPSTR
Address: $EB2B
Symbolic: FPAC1⟶M

Stores value in primary floating point accumulator in memory. Converts from FPAC format to Applesoft variable format. X and Y registers must point at first byte in memory in which value is to be stored (ADL, ADH). Clears $AC.

Name: TR1>2
Address: $EB63
Symbolic: FPAC1

Transfers the value contained in the primary floating point accumulator to the secondary floating point accumulator. Clears $AC.

Name: FPDIV2
Address: $EA60
Symbolic: FPAC2/M⟶FPAC1

Divides the value contained in the secondary floating point accumulator by the value pointed at by the A and Y registers (ADL, ADH) and stores the result in the primary floating point accumulator.

Display Floating Point Representation of Vars.

```
80 :
90 X = 0: D$ =  CHR$ (4)
100  FOR N = 768 TO 792
110  READ A: POKE N, A
120  NEXT
130  REM  ESTABLISH CONVERSION ROUTINE AT $300
140  DATA  165, 105, 24, 105, 2
150  DATA  164, 106, 144, 1, 200
160  DATA  32, 249, 234, 160, 6
170  DATA  185, 157, 0, 153, 25
180  DATA  3, 136, 16, 247, 96
190  HOME : PRINT : PRINT  TAB( 7)"FLOATING
                        POINT CONVERSIONS"
200  PRINT : PRINT : PRINT "INSTRUCTIONS-"
210  PRINT : PRINT "ENTER VALUE YOU WISH
                CONVERTED TO FLOATING POINT
        REPRESENTATION. IF YOU  WISH TO PRINT
                   THE CONVERSIONS ON THE"
220  PRINT "PRINTER, FOLLOW THE VALUE WITH A 'P'.
                     TO RETURN TO BASIC, HI
     T (RETURN) KEY."
230  VTAB 14: CALL  - 868
240  INPUT "ENTER VALUE: "; A$
250  IF A$ = "" THEN  VTAB 23: END : REM
                             ""=NULL $
260  IF  RIGHT$ (A$, 1) >  < "P" THEN 300
270  PRINT D$; "PR#1"
280  REM  PRINTER IN SLOT # 1
290  PRINT : PRINT
300 X =  VAL (A$): CALL 768
310  VTAB 18: CALL  - 958: PRINT "VALUE= "X
320  PRINT : PRINT "ACCUMULATOR:  $";
330  FOR N = 793 TO 799
340 A =  PEEK (N): GOSUB 450
350  NEXT : PRINT : PRINT
360  PRINT "VARIABLE:     $";
370 B =  PEEK (105) +  PEEK (106) * 256 + 2
380  FOR N = B TO B + 4
390 A =  PEEK (N): GOSUB 450
400  NEXT : PRINT
410  PRINT D$; "PR#0"
420  GOTO 230
430 :
440  REM  DECIMAL TO HEX SUB
450 A = A / 16: B =  INT (A)
460 A = (A - B) * 16
470 B = B + 48: IF B > 57 THEN B = B + 7
480  PRINT  CHR$ (B);
490 A = A + 48: IF A > 57 THEN A = A + 7
500  PRINT  CHR$ (A)" ";
510  RETURN
```

Name: TR2>1
Address: $EB53
Symbolic: FPAC2→FPAC1

Transfers the value contained in the primary floating point accumulator to the secondary floating point accumulator. Clears $AC.

Name: FPSQR
Address: $EE8D
Symbolic: FPAC1→FPAC1

Returns the positive square root of the value contained in the primary floating point accumulator in the primary floating point accumulator.

Name: FPEXP
Address: $EE94
Symbolic: FPAC2 M→FPAC1

Raises the value contained in the secondary floating point accumulator to the value pointed at by the A and Y registers. The result is stored in the primary floating point accumulator.

Name: FPINT
Address: $EC23
Symbolic: INT (FPAC1)→FPAC1

Returns the integer value of the value contained in the primary floating point accumulator to the primary floating point accumulator.

Name: FPABS
Address: $EBAF
Symbolic: ABS (FPAC1)→FPAC1

Returns the absolute value of the value contained in the primary floating point accumulator to the primary floating point accumulator.

Name: FPADD
Address: $E7BE
Symbolic: M+FPAC1→FPAC1

Adds the value of the variable pointed to by the A and Y registers (ADL, ADH) to the value contained in the primary floating point accumulator and stores the result in the promary floating point accumulator.

Name: FPADD2
Address: $E7A0
Symbolic: 0.5 + FPAC1→FPAC1

Similar to above, but adds the value (0.5) to the primary floating point accumulator.

Name: FPMUL
Address: $E97F
Symbolic: M*FPAC1→FPAC1

Multiplies the value pointed at by the A and Y registers (ADL, ADH) by the value contained in the primary floating point accumulator and stores the result in the primary floating point accumulator.

Name: FPSUB
Address: $E7A7
Symbolic: M – FPAC1→FPAC1

Subtracts the value contained in the primary floating point accumulator from the value pointed at by the A and Y registers (ADL, ADH) and stores the result in the primary floating point accumulator.

Name: FPDIV
Address: $EA66
Symbolic: M / FPAC1→FPAC1

Divides the value pointed to by the A and Y registers (ADL, ADH) by the value contained in the primary floating point accumulator and stores the result in the primary floating point accumulator.

Name: FPSGN
Address: $EB90
Symbolic: SGN (FPAC1)→FPAC1

Returns the sign of the value contained in the primary floating point accumulator. A negative value will return (−1). A positive value will return a (1). A value of zero will return a (0).

Name: FPLOG
Address: $E941
Symbolic: LOG (FPAC1)→FPAC1

Returns the natural log of the value obtained in the primary floating point accumulator to the primary floating point accumulator.

Name: COMP2
Address: $E89E
Symbolic: TWO'S COMPLEMENT OF FPAC1→FPAC1

Returns the Two's Complement of the value contained in the primary floating point accumulator to the primary floating point accumulator.

Name: INT>FP
Address: $E2F2
Symbolic: (Y,A)→FPAC1

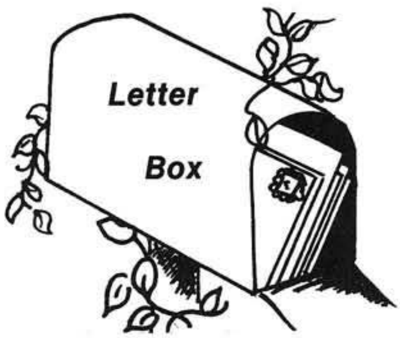
Converts a two byte integer to its floating point equivalent (FPAC format) and stores it in the primary floating point accumulator. The integer must be represented with: the high-order byte stored in the A register, and the low-order byte stored in the Y register.

Name: FP>INT
Address: $E10C
Symbolic: FPAC1→($A0, $A1)

Converts the floating point contained in the primary floating point accumulator to a two byte integer, which is stored in the fourth and fifth bytes of the primary floating point accumulator ($A0, 4A1). $A0 contains the high-order byte and $A1 contains the low-order byte.

μ

R.M. Mottola is a member of the Systems Staff at Cyborg Corp., a manufacturer of medical instrumentation. He is currently involved in the design and enhancement of microcomputer based physiological data acquisition and processing devices.



Dear Micro,

I read the editorial in the December ('79) issue and saw that it said that the first articles written by teenagers were coming soon. Well, I wrote 'An Additional I/O Interface for the PET' (also Dec. '79) and I am 14...

Kevin Erler

Well Kevin, the quality of your article shows how well the teenagers can do — not only in using computers but in writing about them. We did not realize from your article that your were only 14, and we hope to see more works of young writers appearing in Micro.

To the Editor:

I recently ordered a software cassette from Cyberdine, and along with my order I mentioned my difficulty in attaching my recorder to the AIM. Mr. Clark and Peterson phoned me and gave me quick and valuable help. They asked me whether I had the March 1979 AIM Users' Guide. I didn't.

In March '79 I bought the AIM documentation to see whether I wanted to buy the AIM. In December '79 I bought the AIM. I then had two December '78 Users' Guides. I completed and sent to Rockwell the "up-date" postcards at the rear of the books. Results: I have received nothing at all from Rockwell.

This is a shame since I understand that there is much correction and clarification, including a much better and detailed section on how to hook up recorders. (Which I don't have!)

I am certain that your readers with the 12/78 Users' Guides would appreciate your help in getting Rockwell to send them their 3/79 copies — so that they may hook up their recorders.

Edwin Kooser
Flagtown, NJ

We would suggest that you first try writing directly to Rockwell and requesting the specific updated information. It is sometimes difficult for a large company to keep on top of all of the documentation updates, but I am sure that Rockwell will be happy to oblige.

Dear Editor,

I would like to encourage your readers to copy the following letter and send copies to the manufacturers listed below. If enough of us do this, we might see a new, low-cost printer on the market soon.

Thank you,
Bruce Showalter
Abilene, TX

We don't know if Mr. Showalter's method is the best, but he expresses a true need. Here, then, is his letter to manufacturers, and a list of manufacturers that he sent along with it. If you agree with Mr. Showalter, you may want to write to them. We do think that many letters would have some impact, but we suggest that you write your own letters, rather than copy this one, and include your specifications for a printer.

Dear Printer Manufacturer:

As a potential customer, I challenge you to produce a printer with the following features:

* Fixed line capacity of at least 72 characters.
* Fixed spacing of 10 characters per inch.
* At least 83 printable ASCII characters
* Maximum print rate of at least 15 CPS
* 8½-inch forms width capacity
* Price less than $326

By eliminating such frills as multiple line lengths, graphics, and high point speeds, I believe that you can market a printer that doesn't cost more than the microcomputer that drives it.

Please think it over; buyers are waiting!

Yours truly

Axiom
5932 San Fernando Road
Glendale, CA 91202

base 2, inc.
Box 3548
Fullerton, CA 92634

Comprint Inc.
340 E. Middlefield Rd.
Mountain View, CA 94043

Integral Data Systems
14 Tech Circle
Natick, MA 01760

Microtek, Inc.
7844 Convoy Ct.
San Diego, CA 92111

Trendcom
484 Oakmead Pkwy
Sunnyvale, CA 94085

Atari Notes

Some useful information about the Atari 800, including a table of the hex values of the keyboard, a discussion of the string functions, and a demonstration program.

William L. Colsher
4328 Nutmeg Lane, Apt. 111
Lisle, IL 60532

After the initial thrill of owning a new computer has worn off you begin to notice little details that weren't apparent at first. You discover little tricks to make programming easier and little quirks that make it more difficult. This article contains some of the information that I've gathered in the two months I've owned my Atari 800.

Joy sticks are neat. They make writing games with real time motion easier and more fun to play. Unfortunately, you can't assume that every computer owner will have the joysticks your game requires. (The Atari computes can use three different types of game controllers!) In fact, you shouldn't assume that your hypothetical user has anything more than a minimum system.

Looking through the Atari BASIC book that comes with the computer yields only disappointment. There is no BASIC function that just polls the computer keyboard "on the fly". (Like the Radion Shack INKEY$ function.)

Fortunately, there is a way to use the keyboard for real time control. While looking through a program from Sears (!) I discovered just what was needed. It seems that memory location 764 (decimal) contains a value indicating the last key that was pressed. So, by PEEKing that location you can check on what the user is doing at any time without forcing the program to wait with an INPUT statement.

There are, however, two things you have to keep in mind when using this PEEK. First, the value in location 764 does not change until another key is pressed or you poke some new value into it. Second, notice that I've been saying "value". The number at 764 is not an ASCII value of the key pressed. Table 1 contains a list of the values and the corresponding characters.

So, when you use PEEK(764) in a program you have to (1) know the values of the keys you want to use and (2) be sure to reset location 764 after each PEEK. Program 1 is a simple example of using PEEK(764) to control the aiming and firing of a laser turret.

When you CLOAD or CSAVE a program on the Atari computers an internal beeper is triggered once or twice. If you'd like to be able to use that beeper in your programs a PRINT CHR$(253) does the job. I have found it useful to indicate to a user that he's typed something wrong. It's easier to use than the SOUND function since you don't have to turn it off. You can always be sure that it will be heard as well since the SOUND function assumes that the TV set's volume is turned up to an audible level.

Another useful CHR$ number is 125. When you execute a PRINT CHR$(125) the screen is cleared. At first thought this seems a lot like executing a GRAPHICS 0 command. However, when you are using graphics modes other than zero CHR$(125) clears only the text window at the bottom of the screen.

One of the great deficiencies of Atari BASIC is the shortage of string functions. Most of us are used to the Microsoft BASIC functions: MID$, RIGHT$, LEFT$, and concatenation ("+"). Atari BASIC has none of these. Fortunately, it does have enough to allow a programmer to simulate the string functions present in, say, TRS-80 BASIC.

Table 2 contains a summary of the four string manupulation functions mentioned above and their equivalents in Atari BASIC. You should also note that all string variables must be DIMed in Atari BASIC. In the examples A$ and B$ can be DIMed to whatever your application requires but T$, a temporary storage variable, must be DIMed T$(1) so that it has a length of one character.

It should be evident from this article that Atari BASIC is an adequate, if not overwhelming, version of the language. Since the Atari computers use plug in ROM cartridges to hold language interpreters it is possible that Atari will eventually intorduce a more standard version of BASIC. (At this time the only other language available is an Editor/Assembler cartridge.)

Figure 1 has assumed that when using the Atari versions the variables A$ and B$ have been DIMED appropriately. The variables S, E, and L in these examples stand for the Start location of the string, and the End location, and the Length, respectively. μ

Table 1

Key	Value	Key	Value	Key	Value
a	63	s	62	<	54
b	21	t	45	>	55
c	18	u	11	-	14
d	58	v	16	=	15
e	42	w	46	+	6
f	56	x	22	*	7
g	61	y	43	;	2
h	57	z	23	,	32
i	13	1	31	.	34
j	1	2	30	/	38
k	5	3	26	ESC	28
l	0	4	24	TAB	44
m	37	5	29	BS	52
n	35	6	27	CAPS	60
o	8	7	51	RET	12
p	10	8	53	SPACE	33
q	47	9	48	[Atari logo]	39
r	40	0	50		

SHIFT adds 64 to the key pressed.
CTRL adds 128 to the key pressed.
Use 255 to reset location 764. (This is the value it has when the system is powered up or RUN is typed.)

Figure 1:

Concatenation

Microsoft: A$(LEN(A$) + 1) = B$
Atari : A$(LEN(A$) + 1) = B$

LEFT$

**Microsoft: B$** = LEFT$(A$,L)
Atari : B$ = A$(1,L)

RIGHT$

Microsoft: B$ = RIGHT$(A$,L)
Atari : S = LEN(A$)-L
B$ = A$(S)

MID$

Microsoft: B$ = MID$(A$,L)
Atari : E = S + L
B$ = A$(S,E)

Program 1

```
10     GRAPHICS 7+16
15     X=80:Y=95
20     GOSUB 1000
30     REM ***LOOK FOR A KEYPRESS
40     IF PEEK(764)  255 THEN GOSUB 4000
50     GOTO 40
1000 REM ***DRAW LASER TURRET
1010 COLOR 1
1020 SETCOLOR 0,8,8
1030 PLOT X,Y
1040DRAWTO X+4,Y
1050 DRAWTO X+4,Y-4
1060 DRAWTO X,Y-4
1070 DRAWTO X,Y
1080 PLOT X+2,Y-5
1090 PLOT X+2,Y-6
1100 RETURN
2000 REM ***ERASE LASER TURRET
2010 COLOR 0
2020 GOSUB 1030
2040 RETURN
3000 REM ***FIRE LASER
3010 COLOR 2
3020 SETCOLOR 1,3,8
3030 DRAWTO X+2,Y-95
3040 FOR I=1 TO 50:NEXT I
3050 COLOR 0
3060 PLOTX+2,Y-7
3070 DRAWTO X+2,Y-95
3080 RETURN
3100 REM ***MOVE TURRET LEFT
3110 GOSUB 2000
3120 X=X-1
3130 GOSUB 1000
3140 RETURN
3200 REM***MOVE TURRET RIGHT
3210 GOSUB 2000
3220 X=X+1
3230 GOSUB 1000
3240 RETURN
4000 REM ***HANDLE KEYS
4010 REM*** RIGHT AND LEFT ARROWS MOVE
4020 REM ***THE TURRET AND THE SPACE
4030 REM ***BAR FIRES THE LASER
4040 IF PEEK(764)=33 THEN GOSUB 3000
4050 IF PEEK(764)=6 THEN GOSUB 3100
4060 IF PEEK(764)=7 THEN GOSUB 3200
4070 POKE 764,255
4080 RETURN
```

Up From The Basements

by Jeff Beamsley

Hello and welcome to the view of 6502 computing Ohio Scientific style. That isn't meant to exclude 6502 enthusiasts who don't own Ohio Scientific equipment, but OSI users haven't had much of a voice to date. Users of Ohio Scientific machines have long suffered the 'middle child' syndrome of benign neglect from the microcomputing media. I know my customers and dealers have sometimes felt left out of the microcomputing hoopla when Ohio Scientific's name did not show up in articles mentioning Apple, Radio Shack or Commodore. It can get lonely being the only person on the block with an OSI machine. Rest assured though, you OSI users are not alone. As a matter of fact, the number of OSI enthusiasts has reached a point that independent users' groups and newsletters seem to be spontaneously springing up all over the country. Recognizing that trend, and also the fact that Ohio Scientific is a significant force in 6502-based computing, the people at Micro have provided this forum.

One purpose of this column is to get you 'connected' with the Ohio Scientific users' community. If we can't get you connected, maybe can inspire you to start your own club. Whatever you are doing, however, we want to hear from you. Within the space limitations of this column we will certainly attempt to publicize every activity of Ohio Scientific users.

If that sort of publicity were the only purpose of this column, it would be filling a need, but it would not be very interesting for the 6502 enthusiast in general. That would be unfortunate because Ohio Scientific has done some truly remarkable things with the 6502. I could make the arguement that much of what Ohio Scientific has done and is doing represents "state-of-the-art" in 6502-based hardware. They were the first company to offer a completely assembled and tested computer. They were the first company to deliver a machine that had BASIC in ROM. They were the first and remain one of the few companies delivering a microcomputer with a Winchester Technology hard disk. All done with the 6502. They are certainly the only 6502-based computer manufacturer producing machines to compete with the Z-80-based CP/M machines.

Ohio Scientific has grown from a basement operation in Hiram, Ohio to a multi-million dollar company. The story of their growth alone is one of interest to anyone who experienced the microcomputer revolution. They have accomplished all this with a sometimes debilitating, some would say fatal, philosophy that permeates the business side of their activities. Reflections of what could be termed a "basement attitude" give Ohio Scientific a unique image or character in the market place. It very well may have been this image that caused some of you to choose other 6502-based systems. It is this blend of innovative hardware and unusual attitude that I think should prove interesting reading for any 6502 entusiast. The second object of this column, then, will be to discuss new products and innovative design coming from Ohio Scientific and how, in some cases, company philosophy shaped that design.

Another common funcion served by columns such as this is "bug-fix" and "gotcha" information. I am sure that this column will certainly have its share of that. But I hope to take a different approach to it. In many cases, because of the general nature of Ohio Scientific hardware, a "bug-fix" provides an insight into the function of hardware and/or software and may provide an opportunity to experiment and improve that hardware/software rather than just fix it. There is also a lesson to be learned from "gotcha's." Because of the philosophy alluded to in the previous paragraph, many of the gotcha's in Ohio Scientific equipment can be predicted. By examining these product short-comings in that light we can learn how that philosophy works, and hopefully become more intelligent consumers.

I think that Ohio Scientific has a lot to offer, not only in the remarkable hardware that it designs and produces, but also in its history and the way it chooses to do business. I hope that future discussions of these items in this column will prove interesting to all 6502 enthusiasts.

Connections

Publications:
PEEK (65), Aardvark Journal.

Clubs:
OSIO of Ellicott City, MD. Contact: Walace Kendall.
The Challenger News of Philadelphia, PA. Contact: Don Derosa.
Consumer Computers of La Mesa, CA. Contact: Rick Clardy.
ARISTO—Craft of New York, NY. Contact: David Gillette.
Portland Computer Society of Portland, OR. Contact: Roger Giles.

Please Direct all comments, suggestions, questions, etc. to me, Jeff Beamsley, at Tek Aids Industries, Inc., 44 University Drive, Arlington Heights, IL 60004.

Son of Screen Print

This is a 'mini-word processor' which overcomes some of the inherent idiosyncrasies of the Pet Printers.

Kenneth Finn
Little Old Farm
Bedford, NY. 10506

The Commodore Company produces many fine pieces of equipment. Unfortunately, as their business grew the design philosophy changed. Thus later pieces of equipment like the Commodore Printer were designed slightly differently with respect to the Original Commodore PET Computer.

While these changes are not disastrous with respect to equipment compatibility, they produce problems for software designers to mate the equipment.

Here are some examples of this imcompatibility:

1. To produce Letters on the Printer from the screen characters you have to offset the screen ASCII values.
2. To produce lowercase on the printer your have to send a special character every line, vs. a single poke on the Pet Computer.
3. To produce REVERSE characters on the printer you have to send a special character to the printer; the PET uses a single key,in this case RVS ON.
4. To send SHIFTED Characters to the printer you have another offset to change the screen ASCII to printer ASCII.

None of these is very much of a problem in itself but when you try to produce a simple screen print program for the PET PRINTER you have to factor all of these idiosyncrasies into the program.

When I first got the Pet Printer I needed a simple program to produce listings using ONLY letters and numbers. The resultant program (MICRO 22:13) did just that and had the advantage of being usable for other printer like the AXIOM or Selectric typewriters, or any other printer that uses ASCII.

Now this new version is more specialized and tailored just for the PET PRINTERS. Thus it is longer,to get the printer to reproduce the entire PET graphics and lowercase character set. This program is really a mini-word processor in that you can get a "letter" on the screen correct it with the cursor and then send it to the printer. The same can be done with pictures etc.

I have tried to document the program well so you can study it and learn from the approach. It is similar to the previous program but the addresses were changed for the new ROM set and the character conversions are more specialized.

To use the program use SYS 826. If you want to reproduce the entire screen change $036E to $19. Then, if you are in a BASIC program you can have the printer reproduce the screen with a simple sys command. The program fits into the second cassette buffer and will remain through many new program loadings. You can even load it after a BASIC program to use when you need it.

μ

Kenneth Finn has a B.S Degree in Electrical Engineering and a Business Degree in Organization Behavior. His interest in computing started when he had to analyze employee attitude survey data from his consulting clients. His pet is still used for this purpose but is also used extensively in word processing and he is completing an interface with an IBM Selectric Type 735. He also has a business venture training managers in the use of micro computers for analysis models of business plans, using the PET computer exclusively because of its simple, one piece type of operation.

Old PET Hexadecimal addresses

```
033A  A9 00     LDA #$00
033C  8D 65 02  STA $ 0265   $0256   Pet Printer Secondary Address =0
033F  8D FE 03  STA $ 03FE           Line Counter
0342  8D FF 03  STA $ 03FF           Characters per line counter
0345  85 DA     STA $ DA             Lo-Byte Index
0347  A9 01     LDA #$01
0349  8D AE 00  STA $ 00AE   $0262   GP1B File length =1
034C  A9 04     LDA #$04
034E  8D 51 02  STA $ 0251   $0242   Logical File Number Four
0351  8D 5B 02  STA $ 025B   $024C   Device Number Four
0354  A2 04     LDX #$04
0356  20 BC F7  JSR $ F7BC   $FFC9   Open 4,4,0
0359  A9 80     LDA #$80
035B  85 DB     STA $ DB             Hi-Byte Index
035D  20 C6 03  JSR $ 03C6           Test for PET Lowercase Mode
0360  AD FF 03  LDA $ 03FF
0363  C9 28     CMP #$28             40 Characters per line constant
0365  D0 17     BNE $ 037E
0367  EE FE 03  INC $ 03FE
036A  AD FE 03  LDA $ 03FE
036D  C9 15     CMP #$15             21 Lines per screen constant
036F  F0 4A     BEQ $ 03BB           (End Program return to basic)
0371  A9 00     LDA #$00
0373  8D FF 03  STA $ 03FF
0376  A9 0D     LDA #$0D             Carriage Return  Chr$(13)
0378  20 32 F2  JSR $ F232   $F230   Print#4
037B  20 C6 03  JSR $ 03C6           Test for Pet Lowercase Mode
037E  38        SEC
037F  A2 00     LDX #$00
0381  A1 DA     LDA ($DA,X)          Get Next Character via indirect ad-
                                     dress
0383  C9 20     CMP #$20             Is it a letter? then add 64
0385  90 1B     BCC $ 03A2
0387  C9 40     CMP #$40             Is it a Number?  then add 0
0389  90 1D     BCC $ 03A8
038B  C9 60     CMP #$60             Is it shifted letter then add 128
038D  90 17     BCC $ 03A6
038F  C9 80     CMP #$80             Is it shifted number then add 128
0391  90 13     BCC $ 03A6
0393  85 DC     STA $ DC             Temporary Storage
0395  A9 12     LDA #$12             Pet Printer Reverse On Chr$(18)
0397  20 32 F2  JSR $ F232   $F230   Print #4
039A  A5 DC     LDA $ DC             Re-Load Temporary
039C  38        SEC
039D  E9 80     SBC #$80             Subtract 128 for reverse characters
039F  38        SEC
03A0  B0 E1     BCS $ 0383           (Unconditional Branch)
03A2  69 40     ADC #$40             Offset for letters
03A4  10 02     BPL $ 03A8
03A6  69 80     ADC #$80             Offset for shifted characters
03A8  20 32 F2  JSR $ F232   $F230   Print #4
03AB  A9 92     LDA #$92             Turn off Per Printer Rvs. chr$(146)
03AD  20 32 F2  JSR $ F232   $F230   Print #4
03B0  E6 DA     INC $ DA             Index Lo-byte
```

```
03B2  D0 02       BNE  $ 03B6
03B4  E6 DB       INC  $ DB                Index Hi-byte
03B6  EE FF 03    INC  $ 03FF              Increment character counter
03B9  10 A5       BPL  $ 0360              Return for next character
03BB  A9 0D       LDA  #$0D                ASCII Carriage Return
03BD  20 32 F2    JSR  $ F232     $F230    Print #4
03C0  A9 04       LDA  #$04
03C2  20 6E F2    JSR  $ F26E     $FFCC    Close 4
03C5  60          RTS                      Return to basic
03C6  AD 4C E8    LDA  $ E84C              Pet Graphics shift on Port
03C9  29 02       AND  #$02                Get bit 2
03CB  C9 02       CMP  #$02                Test for lowercase
03CD  D0 05       BNE  $ 03D4
03CF  A9 11       LDA  #$11                Pet Printer Lowercase Chr$(17)
03D1  20 32 F2    JSR  $ F232     $F230    Print #4
03D4  60          RTS                      Return
```

Classified Ads

Business Dollars and Sense in Applesoft

If you ever intend to do serious business programming in BASIC, then the information and programs presented here are invaluable. They show how to overcome the inherent rounding and formatting problems of BASIC in dealing with dollar and cents type of data.

Barton M Bauers, Jr.
30 Hillock Dr.
Wallingford, CT. 06492

If you purchased an Apple II Plus for business applications, that is, applications that require the use of financial tables and calculations, then you may have encountered a rounding problem in executing your programs. Perhaps you have failed to recognize this problem, and are running programs which contain erroneous mathematical calculations! The purpose of this article is to acquaint you with the potential for rounding errors, and to suggest several possible solutions, depending on your needs. In addition, the process of creating textfiles, with some examples, will be addressed, since you will probably wish to use the subroutines discussed later in many programs which you write.

To start, let's demonstrate the problem. Try the following program:

PRINT 100.09 + 200.00 + .80 (rtn)

(Note that where (rtn) is indicated, it means to press the key marked RETURN.)
Your Apple should display:

300.89

Now type this program:

PRINT 300.89 — 100.09 — 200.00 — .80 (rtn)

The answer (which you'll agree should be zero) will appear as:

1.19907782E-08

This small error occurs because not all numbers between zero and one can be exactly represented in binary arithemetic. Oddly enough, for most scientific work, such an error is insignificant, and will not affect the outcome of any programs. It is unlikely, however, that any usable system can be inplemented in a business or financial situation unless absolute accuracy is obtained in recording and tabulating monetary amounts. When you program such an application — whether it be the family checkbook, or a complicated inventory control system — the ability to balance to the penny is a must!

There is, fortunately, a straightforward answer to the problem. While it is easy to discuss, it requires a bit of trickery to implement. If all values are carried within the computer as whole (integer) numbers, then there is no possibility of having roundng errors. The sacrifice you make, of course, is the necessity of performing all internal mathematical calculations in whole numbers, which requires that you, the programmer, remember where the decimal point belongs. Basically, therefore, by multiplying each monetary value by 100, and taking the INTeger value of the resultant figure, the problem is solved. This opens up additional problems, as we shall see.

Type in the following program:

```
  10 DEF FN VL(X) = INT(X* 100)
  20 INPUT "ENTER NUMBER:"; K
  30 K = FN VL(K)
  40 PRINT "NUMBER IS NOW: ";K
  50 GOTO 20
RUN
```

Try some of the following examples:

1.00 (rtn)

The computer will respond with -

100

Now try this one:

-2.99 (rtn)

The Apple answers with

-300

OOPS! Try this one now-

300.89

Your answer:

30088

Clearly, the use of integer values does not in itself solve the problem. The same rounding error which plaged the initial examples is contained in the integer value. The library function INT supplies the "...largest integer less than or equal to the given argument..." (quoted from the Applesoft II manual). In the negative direction, the rounding error will cause the integer value to one number smaller (further negative) than the argument whenever there is a rounding error: in the positive direction the integer is similarly smaller when the computer underrounds.

Referring back to the example used at the beginning of this article, it is easy to see that the value of the rounding error is extremely small, something like .00000001. Using the integer approach to eliminate the rounding problems, then, requires consideration for his small error. We are not concerned with values smaller than the second decimal place (pennies) in about 98% of business applications, therefore it is possible to add enough "cushion" to the integer conversion routine such that the small error which creeps in will never cause the Applesoft command INT to fall short during conversion.

To illustrate this process, type CTRL C (rtn) and rekey line 10 as follows;

```
10 DEF FN VL(X) = INT((X + .0001) * 100)
RUN (rtn)
```

Now try entering the previous examples.

Number Entered	Value Returned
1.00	100
-2.99	-299
300.89	30089

This function works for both positive and negative numbers, because the 'adder' of .0001 is enough to offset any internal underrounding, both in a positive and a negative direction. Therefore, in any problem involving money calculations, you should add the following to your program:

```
15 DEF FN VL(X) = INT ((X +.0001) * 100)
          .
          .
          .
aaa INPUT "ENTER AMOUNT";C

bbb C = FN VL(C)
          .
          .
          .
```

Line 15 defines the function;

Line aaa requires keyboard entry of an amount which will be stored as variable C internally (you will naturally use what ever variable name you need here);

Line bbb converts C to an integer value, using the previously defined function, and 'pads' the value read in before conversion to prevent underrounding.

Remember — all internal mathematics must now be performed with whole numbers.

A natural question at this point would be, "How do I print out the figures so that they once again look like dollars and cents?" This is part two of our story.

It would seem that by multiplying the integer number to a decimal number similar to the one originally typed in. Try it!

Type the following -

PRINT 30089* .01 (rtn)

Your answer:

300.89

```
                    *** MASK ***
15 DEF  FN VL(X) =  INT ((X + .0001) * 100)
15000  REM   SUBROUTINE MASK : REM ARGUMENT IS ZZ$ : REM RESPONSE IS XW$
15010 M% =  LEN (ZZ$)
15020 XU$ = "$":XX$ = "":XY$ = "."
15030 XZ$ =  RIGHT$ (ZZ$,2)
15040  ON M% GOTO 15060,15070,15100,15100,15100,15100,15100,15100,15100
15050  PRINT "ERROR ON INPUT VALUE ": GOTO 15120
15060 XZ$ = "0" + XZ$: GOTO 15110
15070  IF  LEFT$ (ZZ$,1) = "-" GOTO 15090
15080  GOTO 15110
15090 XZ$ = "0" +  RIGHT$ (XZ$,1):XX$ = "-": GOTO 15110
15100 XX$ =  LEFT$ (ZZ$,(M% - 2))
15110 XW$ = XU$ + XX$ + XY$ + XZ$
15120  RETURN
```

Listing 1

```
                    *** CREATE EXEC FILE ***
63999 D$ =  CHR$ (4): INPUT "NAME OF TEXTFILE IS - ";AA$: PRINT D$;"OPEN "
        ;AA$: PRINT D$;"WRITE ";AA$: LIST 1,63998: PRINT D$
        ;"CLOSE ";AA$: DEL 63999,63999
```

Listing 2

```
                    *** CHECK PROTECT ***
15500  REM   SUBROUTINE CHECKPROTECT : REM ARGUMENT IS ZZ$ : REM RESPONSE IS XW$
15510  IF  LEFT$ (ZZ$,1) = "-" GOTO 15560
15520 M% =  LEN (ZZ$)
15530 XU$ = "$":XY$ = "."
15540 XZ$ =  RIGHT$ (ZZ$,2)
15550  ON M% GOTO 15570,15600,15580,15580,15580,15580,15580,15580,15580
15560  PRINT "ERROR ON INPUT VALUE":XW$ = "": GOTO 15660
15570 XZ$ = "0" + XZ$: GOTO 15600
15580 XX$ =  LEFT$ (ZZ$,(M% - 2))
15590  GOTO 15620
15600 XW$ = XU$ + XY$ + XZ$
15610  GOTO 15630
15620 XW$ = XU$ + XX$ + XY$ + XZ$
15630 XT$ = "******************************":B = 30 -  LEN (XW$)
15640 XS$ =  RIGHT$ (XT$,B)
15650 XW$ = XS$ + XW$
15660  RETURN
```

Listing 3

Try some additional values.

Value	Value * .01
-299	-2.99
-100	-1
180	1.8

Again, the result is unacceptable for business applications. Again it is clear that Applesoft BASIC, which handles scientific applications so well, is not equipped to yield usable formatting in dollars and cents. The author in fact, has seen commercial software which ignores this problem, and gives answers with the same errors demonstrated throughout the article. While some programmers might not consider the rounding problem serious, how can a businessman issue a check for $1.8?

The answer to the problem of restoring two decimal places to the internally generated integer values is a program which is named subroutine MASK. This program should be typed and saved, converted to a textfile, and exec'd into every business application where accurate dollars and cents calculations are required. Listing 1 shows the program steps for MASK. Type it and save it under the name DOLLAR MASK (it is assumed that you have at least one disk drive). After it is SAVEd, you are ready to make a textfile out of DOLLAR MASK. To do this, if you have not already created a utility program for making textfiles, there is another short program which must be typed, SAVEd, and made into a textfile. Prior to that exercise, however, let's look at the contents of the program MASK.

Line 15 is the value conversion function described earlier.

Line 15010 establishes the number of digits in the variable.

Line 15030 takes the right two characters (cents) and puts them in string variable XZ$. Note however that line 15060 puts a zero ahead of the value stored in XZ$ if XZ$ contains only one digit. Line 15090 removes a minus sign if it became embedded in XZ$, and replaces it with a zero, moving the minus sign to the left of the decimal point in XX$.

Line 15040 branches depending on whether the input string ZZ$ has 1,2, or 3-9 digits.

Line 15100 puts all except the cents value (which is now stored in ZZ$) into the 'dollars' area, XX$.

To test this program, load it from the disk, and add the following additional lines:

```
20 INPUT "ENTER NUMBER: ";CA
30 CA = FN VL(CA)
40 ZZ$ =STR$(CA)
50 GOSUB 15000
60 CA$ = XW$
70 PRINT " THE ANSWER IS : ";CA$
80 END
```

Now type RUN and try some values which might be representative of a business application. Try some positive and negative values, so you can demonstrate that DOLLAR MASK really works.

After you have become familiar with the logic, it is easy to add other capabilities to the DOLLAR MASK. For example, if you want to remove the floating dollar sign from the program, delete the first part of line 15020, and drop XV$ from line 15110. Another example is shown in Listing 3, a routine for adding check protecting characters (*) to the left of the masked number. The asumption in this subroutine is for a field of 30 digits, but you can easily increase or reduce it at your leisure.

To put the finishing touches on your program, it will be necessary to convert DOLLAR MASK into a textfile. Then, it can be added to any program you write by typing EXEC MASK. If you are not comfortable with the EXEC portion of the Apple DOS manual, then the program listed in Listing 2 will do the job easily. To use this program, follow these steps:

1.Type the program in listing 2 TWICE, once with line number 10, and once with line number 63999. When typing it under line number 10, change the LIST reference to LIST 63999.

2.Type RUN.

3. The computer will ask NAME OF TEXTFILE — , to which you should respond CREATE EXEC FILE (rtn). When the disk stops, you will have created a textfile named CREATE EXEC FILE. LOCK it, since it will permit you to set up standard subroutines as text files in the future.

Now you are ready to make DOLLAR MASK into a textfile. If you have already typed it and SAVEd it to disk under the name DOLLAR MASK, LOAD it into memory, and follow the steps below:

1. Type EXEC CREATE EXEC FILE

2. Type RUN 63999

3. Type Answer the inquiry with MASK (rtn)

4. You now have subroutine MASK stored on disk for future use.

Below is a summary on how to get MASK into your future business programs:

1.When writing a program do not use line numbers 15 or 15000 to 15120.

2. Insert the disk with MASK on it and type EXEC MASK.

3. You now have the subroutine and the function in your program

4. Each time your program requires a value from the keyboard such as, for example, CA, add the following line after you read the value in:

```
CA =FN VL(CA)
```

5. If you have occasion to output money data to the screen or to a printer, add the lines:

```
ZZ$ =STR$(CA)
GOSUB 15000
CA$ =XW$
PRINT CA$
```

6. You now have a string variable CA$ to display the value previously stored in CA as a whole number.

7.Remember — the argument to use before you GOSUB 15000 is ZZ$, and the return argument is XW$

μ

Barton M. Bauers is the Executive Vice President of LFE Corporation, Fluids Control Division. His programming background includes Fortran, PL-I, and Basic. Mr. Bauers holds a Master of Science Degree in Industrial Engineering with a concentration in Operations Research.

BCD Input To A 6502 Microprocessor

Many laboratory devices output data to displays in Binary Coded Decimal (BCD) format. Some techniques are presented for interfacing such devices to 6502 based systems, and a BASIC conversion program is provided.

Richard Soltero
Ciba-Geigy Corp.
556 Morris Ave.
Summit, NJ. 07901

In most scientific and medical laboratories there is a proliferation of analytical instrumentation and equipment. In past years these instruments have used D.C. analog signals for transmission of their data to strip chart recorders. With the development of resident microprocessor chips in newer instruments, on-board analog to digital converters (ADC) have become very popular. This innovation allows the use of digital panel meters (DPM) for readouts and for BCD (Binary Coded Decimal) output. Typically, four panel meters are used for displaying the results from the analytical instrument so that values in the range -9999 to +9999 (the leftmost meter has a ± sign) can be detected. The obvious advantages of these developments are readability and the capability of recording data with remote digital printers or plotters. The BCD lines used in these applications can easily be tapped to permit transferring data to a microprocessor and, in many cases, the newer instruments are supplied with a BCD output connector.

The sixteen input lines of a 6522 VIA are ideally suited for transmission of parallel BCD (Binary Coded Decimal) information into a microproceser such as the SYM-1 or AIM-65. In our laboratory we have several analytical instruments with BCD output and are applying this technique for data collection and concentration. The hardware connections are relatively easy requiring only minimal proficiency with a soldering iron. Programming was also simple since BASIC can be used as the programming language throughout.

Our initial application was to collect data from the BCD output connector of a Beckman U.V.-Vis. spectrophotometer (Model 25-7). This is equipped with seven spectrophotometric cells which can be sequentially placed in the light path of the instrument. As light passes through the cell, it is absorbed by the compound of interest. Since the absorbance by a compound in a solution is proportional to its concentration, this technique is useful for quantitive analysis. The absorbance of the solutions in each cell is shown on the digital panel meters of the instrument as each cell is positioned in the light path of the instrument. This digital information is simultaneously made available on the pins of the BCD output connector. During the course of our experiments, the solutions in the cells are constantly changing and the absorbance values, determined by the instrument, can be transmitted to the microprocessor and stored in RAM.

The data is arranged on the BCD output connector of the instrument so that each digit is represented by four bits. In a four digit connector these digits are the Units (U), Tens (T), Hundreds (H) and Thousands (K). The total picture for the BCD connector is:

Port A U(1) U(2) U(4) U(8) T(1) T(2) T(4) T(8)
Port B H(1) H(2) H(4) H(8) K(1) K(2) K(4) K(8)

Each of these 16 lines can be hard wired to one of the input pins of Ports A and B of the 6522 VIA, i.e. U(1) = Port A, Bit 0; U(2) = Port A, Bit 1; H(1) = Port B, Bit 0; H(2) = Port B, Bit 1, ect. On the A connector of the SYM-1, PB6 is used for a keyboard function, however, all 16 pins are available from the user supplied VIA on the AA connector.(All 16 lines on the A connector of an AIM-65 are available).

The values that appear at these ports are PEEK'd into memory using a BASIC program. As will be shown in the sample progaram, the (2.4-7.0 VDC) voltages appearing on the VIA pins are the BCD representation of what is displayed on the digital panal meters of the instrument. If the Hex keyboard and display on the SYM-1 are used to look at this data, it will be identical to the digital panel meters. If the AA connector is used, memory locations $A801 will contain the low order digits and $A800 will contain the high order digits.

When the values are PEEK'd into BASIC, all hex numbers are converted into decimal numbers, and memory locations $A800 and $A801 become locations designated as 43008 and 43009, respectively. The BCD data also becomes a decimal representation after a conversion from hexidecimal. It becomes necesary at this point to convert the BCD data back to its original value since it is already decimalized when it appears on the input pins of the VIA.

The conversion of each pair of

high and low order digits is done in subroutine 500. Initially the data is PEEK'd in and it is assigned to a BASIC variable. The variable is carried to the subroutine where the hexadecimal representation of the data in converted to binary digits by a typical algorithm. In step 560, the binary digits are translated into the decimal number and this value is returned to the main line program.

The low order digits are done first and then the high order digits. When they have been returned, the high order digits are multiplied by 100 and added to the low order digits ; the BCD number is now in its original representation. This value is stored in an array for calculations later in the program.

The BCD input data from the instrument is always on the VIA pins and changes as the digital panel meters change. Since it is impractical to collect data constantly, a method was devised to select the required data. The instrument sends out a 28 VDC signal which is dropped to ground each time a new cell is brought into position for an absorbance reading. This voltage is stepped down to 5 VCD signal which is dropped to ground each time a new cell is brought into position for an absorbance reading. This voltage is stepped down to 5 VCD using an optical coupler and is connected to the pin from Port A, Bit 0 on the A connector.(In BASIC, Port A is at location 40961). When a cell is changed, this bit stays low for about 0.5 seconds. The BASIC program stays in a loop which PEEKs in the value of the input register A (IRA) each time through the loop (Program steps 50 to 100). When the 28 VDC is dropped to ground, a 0 appears in Bit 0 of this register, and the program break out of the loop and is sent into a 2 second delay subroutine.

This gives the digital panel meters a chance to stablize before the program returns and Peeks in the BCD values that are represented on the DPM's.

An external push button is connected between Port A, Bit 1 and the ground of the microprocessor. When this button is pushed, the circuit is completed and BIT 1 is forced to zero. The program will PEEK in the values of ;the IRA until it changes. As seen above, when Bit 0 goes to 0, the program goes into a data collection routine. Forcing Bit 1 to zero is an external method of indicating to the processor that the experimental and data collection has finished.

The data collected in this experiment has been formatted into a two dimensional array, (7 x 30). There are seven cells in the instrument and these can be read up to 30 times. The SYM-1 in this set up is a 4K version and restricts us to this data array size. There is an editing section in subroutine 600 where data can be corrected if this is necesary. After some input of instructions at step 300, a simple ratio calculation is done for each data point and the resultant percentages are printed. These final steps are a demonstration of how data can be treated by a program, although many other possibilities exsist such as having this data stored permanently on tape cassettes or having it transmitted directly to a larger main frame computer which can handle the data in much greater detail and can output formal reports of the experiment and the data.

This experiment and the data collection is a relatively simple application of the potential of the microprocessor. The BCD data transmission does away with the necessity of dealing with analog signals which would require the use of an A to D convertor, and the use of BASIC greatly simplifies the programming tasks. It is hoped that this procedure will pave the way for the efficient and practical solution of future instrumental and data collection problems.

μ

Continued from page 15.

In the world of microcomputing,
There is one fact, not worth disputing.
When Micro is read,
It goes to your head,
And your programs begin executing.

D. Duckworth
Las Vegas, NV

A commuter tutored her suitor
To program her microcomputer.
He eloped with the terminal
And her Micro journal
Because the tutor's computer was cuter.

H.I. Mathis
Southfield, MI

There you have the ten best entries of the 1980 Micro Limerick Contest. We agree that it is tough to narrow the best down to one, but please don't delay in voting for your favorite. We hope for even more votes than limerick entries.

Once again, we thank all who entered the contest, and we wish all of the "best ten" the very best of luck.

Send your vote for the best limerick to:

MICRO Limerick Vote
Box 6502
Chelmsford, MA 01824

```
10 PRINT"BCD INPUT PROGRAM (25-7)"
20 DIM L$(8),L(7,30),B(8),R(6)
30 PRINT"INPUT EXPERIMENT #":INPUT L$:PRINT
40 PRINT"EXPERIMENT # ";L$;TAB(30);"ABSORBANCE"
50 REM INPUT SECTION APA0=DATA READY, APA1=END OF RUN BEQ
55 PRINT"CELL #          1        2        3        4        5        6        7"
57 PRINT:CYCLE =1:PRINT"CYCLE #"
60 FOR N=1 TO 7:GOSUB 650:REM DELAY 2 SECS
70 IF PEEK(40961)=254 THEN 200:REM WAIT FOR CELL CHANGE
80 IF PEEK(40961)<254 THEN 300:REM END  OF RUN SIGNAL
90 GO TO 70:REM CONTINUE LOOPING UNTIL A BIT IS CLEARED
100 GOSUB 650:REM DELAY 2 SECS
200 REM PEEK DATA IN, CONVERT TO DECIMAL, STORE IN ARRAY
210 Z=PEEK(43009):GOSUB 500:REM LSD TO DECIMAL
220 LSD=Z:Z=PEEK(43008):GOSUB 500:REM MSD TO DECIMAL
225 L(N,CYCLE)=(LSD+Z*100)*100
240 NEXT N
245 C=CYCLE
250 REM PRINTDATA FROM CURRENT CYCLE
260 PRINTC;TAB(15);L(1,C);L(2,C);L(3,C);L(4,C);L(5,C);L(6,C);L(7,C)
270 CYCLE=CYCLE+1
280 GO TO 60:REM RETURN TO INPUT LOOP
290 REM
300 REM END OF RUN DETECTED
310 PRINT"EXPERIMENT #";L$;" COMPLETED":PRINT"NEED CORRECTIONS TO DATA?

320 INPUT Y$:IF Y$="Y" THENGOSUB 600:REM GO TO EDITING SUBROUTINE
325 K=1:REM DATA CALCULATION SECTION
330 PRINT"ENTER END OF RUN AND RAPID STIR CYCLE #'S":INPUT ER,RS
335 PRINT:PRINT:PRINT"EXPERIMENT #";L$;"  CALCULATED % RELEASED"
340 K=K+1:REM CYCLE COUNTER
345 PRINT"CYCLE # ";K
350 FOR N=1 TO 6
360 IF L(N,RS)-L(N,1)<=0 THEN R(N)=0
370 IF L(N,RS)-L(N,1)<=0 THEN 390
380 R(N)=)L(N,K)-L(N,1))/(L(N,RS)-L(N,1))*100
390 NEXT N
400 PRINT"CYCLE #          1        2        3        4        5
6"
410 IF K<ER THEN 340
420 GO TO 700
485 PRINTN;TAB(13);R(1);TAB(22);R(2);TAB(31);R(3);
486 PRINT TAB(40);R(4);TAB(49);R(5);TAB(58);R(6)
500 REM PEEKED DATA TO BINARY TO DECIMAL
510 FOR D=1 TO 8:B(D)=0:NEXT D
520 FOR D=1 TO 8:X=Z/2:Y=X AND 32767
530 IF X>Y THEN B(D)=1
540 IF Y=0 THEN 560
550 Z=Y:NEXT D
560 Z=B(1)+B(2)*2+B(3)*4+B(4)*8+(10*(B(5)+B(6)*2+B(7)*4+B(8)*8))
570 RETURN
580 REM
600 REM EDITING SUBROUNTINE
610 PRINT"ENTER CELL #, CYCLE #, DATA":INPUT E,F,G
620 L(E,F)=G:PRINT"ANY MORE?":INPUT Y$
630 IF Y$="Y" THEN 610
640 RETURN
650 FOR COUNT=1 TO 750:REM DELAY 2 SEC
660 NEXT COUNT
670 RETURN
700 END
```

The MICRO Software Catalog: XXIII

Software announcements for the 6502 based systems

Mike Rowe
P.O. Box 6502
Chelmsford, MA 01824

Name: **Death Run**
System: **Apple II, ITT 2020**
Memory: **8K**
Language: **Apple Soft**
Hardware: **Apple II**

Fast moving real time version of last few mins of Starwars. Can you pilot your x-wing along and then into the canyon while skillfully avoiding TIE's firing laser bolts? And will your BOMB be shot before dropping into the exhaust and destroying the Death Star?

Copies: **Just released**
Price: **Listing $6.00**
Cassette $7.00
Author: **Stephen Owens**
Available: Stephen Owens
19 Wadeson Way Croft
Warrington, Cheshire
England

Name: **Super Artillery**
System: **Apple II**
Memory: **16K ROM Applesoft**
or 48K RAM Applesoft
Hardware: **Apple II, Disk II**

Fast version of Artillery. It plays with two players, keeps score, sound effects, and other options. You and your opponent will battle it out by shooting at each other by entering angle and velocity of your missile while compensating for the mountain terrain and wind factor. Mountain profiles and missile trajectories are plotted in HI-RES graphics.

Copies: **Just released**
Price: **$20.00**
Author: **Greg Stein**
Available: Rainy City Software
4360 SW Parkview
Portland, Oregon
97225

Name: **Commodity File**
System: **Apple II, Apple II Plus**
Memory: **32K, Applesoft ROM**
48K, Applesoft RAM
Language: **Applesoft II**
Hardware: **Disk II, 132 column printer (optional)**

Description: Stores and retrieves virtually every commodity traded on all Future's exchanges. A self-prompting program allowing the user to enter short/long contracts. Computes gross and net profits/losses, and maintains a running cash balance. Takes into account any amending of cash balances such as new deposits or withdrawals from the account. Instantaneous readouts (CRT or printer) of contracts on file, cash balances, P/L statement. Includes color bar graphs depicting cumulative and individual transactions. Also includes routine to proofread contracts before filing.

Price: **$19.95 plus $2.00 p&h**
Terms: Check or money order
Includes: Diskette, documentation
Available: Mind Machine, Inc.
31 Woodhollow Lane
Huntington, New York
11743

Name: **Baudet Printer Driver Routine**
System: **Apple II**
Memory: **16K**
Language: **Assembly**

Description: Less than ½ K. Handles all special characters. Use to drive inexpensive teletype such as model 15, 19, 28, etc.

Copies: **New**
Price: **$7.00**
Author: **A.B. Buscaglia**
Available: A.B. Buscaglia
2497 W. River Road
Grand Island, N.Y.
14072

Name: **Amateur Radio Oscar Orbital Predictions**
System: **Apple II**
Memory: **16K**
Language: **Applesoft II**

Description: Oscar 7 and 8. Lists data for each orbit for desired day. Displays antenna beam, azimuth and elevation data for specified qth. Can be used with Baudot Driver routine.

Copies: **New**
Price: **$7.00**
Author: **A.B. Buscaglia**
Available: A.B. Buscaglia
2497 W. River Road
Grand Island, N.Y.
14072

Name: **The Relationship Life Dynamic**
System: **Apple II**
Memory: **48K**
Language: **Applesoft, Machine**
Hardware: **Apple II Plus, Disk II**

Description: Now you can use your computer to help make your relationships work. The disk includes an elevator which you control as you make choices about challenges in relationships. Includes animation and a special game called Relatopoly! Now you can circumvent trips like est and Lifespring, save money, and experience transformation in your relationships using this program at your convenience. The second disk in our Life Dynamic Series (of 11 disks). Send for information.

Copies:	**Many**
Price:	**$15.95**
Includes:	Disk, instructions
Available:	Avant-Garde Creations P.O.Box 30161 Dept.MCC Eugene, OR 97403

Name:	**CHAT (Challenger Terminal)**
System:	**OSI Challenger 1P and Superboard II**
Memory:	**4k**
Language:	**Machine code, Basic**
Hardware:	**Modem and RS-232 modification**

Description: An intelligent terminal with the ability to directly transmit data from cassette and transfer received data to cassette via an internal buffer. This buffer automatically expands on systems with more than 4k to allow more data storage at a single time. Full/half duplex modes; selectable parity and stop bits. Chat has a very unique feature — 46 user definable 6-state keys capable of generating all ASCII characters. Standard ASCII keyboard layout. All key changes are stored when CHAT is saved on tape. Also, the keyboard has the auto-repeat feature and a break key.

Price:	**$24.95**
Includes:	Cassette, manual
Author:	**Charles A. Shartsis**
Available:	Charles A. Shartsis 9308 Cherry Hill Road College Park, MD 20740

Name:	**Flexipay**
System:	**Apple II**
Memory:	**48K ROM, RAM Applesoft**
Language:	**Applesoft**
Hardware:	**Apple II, Disk II, Serial interface printer**

Description: A versatile payroll system. Payroll masterfiles for any number of companies may be created, edited and fully maintained. Processes weekly, monthly, salaried, hourly, commissions, etc. Automatic group insurance deductions plus two other auto deducts (credit union, etc.) of your choice. All taxes computed. Pay and nonpay adjustments (advances, etc.). Output includes masterfile, payroll summary with current, WTD and YTD data for each employee, company totals. Labor summary, tax summary. Formatted checks and stubs available. MANY MORE FEATURES!

Price:	**$75.00 Disk plus $2.00 handling** TX residents add 5 percent sales tax. **Sample output $3.00ppd.**
Includes:	System disk, instructions
Author:	**S. Prater, CPA**
Available:	INDATA INC. 8222 Antoine, Suite 103 Houston, TX 77088

Name:	**Tiny Pascal**
System:	**Pet 16K/32K, Apple II w/DOS**
Memory:	**Pet-16K/32K New ROMS, Apple 32K A.S. ROM 48K A.S. RAM**
Language:	**Basic and Assembly**
Hardware:	**Pet cassette or disk, Apple II disk only**

Description: Complete package to create, compile and execute programs that are written in the TINY PASCAL language. Includes line editor for source program maintainance; Compiler to translate source to P-code and Interpreter for execution.

Copies:	**Just released**
Price:	**cassette version $40.00, diskette version $35.00**
Author:	**Arnie Lee, Norm Draper**
Available:	ABACUS Software P.O. Box 7211 Grand Rapids, MI 49510

Name:	**Soft-Sonic**
System:	**Apple II (or Plus except for speech program)**
Memory:	**32k**
Language:	**SS Assembly** **Home control, timing-Applesoft** **Home control, speech-Integer**
Hardware:	**Home control, speech - heuristics speech lab, Ultra sonic transducer and cable (included with programs) Disk highly recommended**

Description: A collection of programs to provide for home control using a BSR or Sears ultrasonic command console. SS is a relocatable machine language subroutine that produces all the codes, tones, and delays required to communicate with the BSR. Home control, Speech provides for verbal control of up to ten lights and appliances with vocabularies for two persons automatically saved and exchanged from disk. Internal software clock; allows for several hundred NAMED sequences. Much more.

Price:	**$39.95 plus $2.00 s&h** GA residents add 4 percent sales tax.
Includes:	SS, Home Control (Speech), Home Control (Timing), SS-Relocator, all on disk (tape by request) and the ultra-sonic transducer complete with cable.
Author:	**John Blankenship**
Available:	B.A.C.E. P.O.Box 52785 Atlanta, Ga 30355

Name:	**Data Factory 2.2**
System:	**Apple 1 or 2 disk Drives, or Hard Disk (Corvus.Lobo) optional printer**
Memory:	**48k RAM, ROM card, Language Card**
Language:	Applesoft

Description: A data base file program of unique utility. It allows the user to create a file consisting of desired catagories (columns or fields) in which various sorting and printing procedures can be accomplished. The program can be copied and lists for user modification. It uses either one or two disk drives and operates with or without a printer. A unique feature of this program is it's construct append features that allows the user to reformat and structure the file names, locations, lengths, and order after data has been entered. Many features!

Price:	**$100.00** (Hard disk version slightly more)
Includes:	Disk, program and manual
Author:	**William Passauer**
Available:	Andent Inc. 1000 North Avenue Waukegan, IL 60085

Name: **PBASIC-DS**
System: **Apple II**
Memory: **32K ROM, 48K Applesoft**
Language: **Applesoft**
Hardware: **Disk**

Description: A preprocessor for Integer and Applesoft programs. Programs are written using a mixture of BASIC statements and pseudo-statements that facilitate the implementation of structured logic. Nine pseudo-statements are supported, including IF...ELSE...ENDIF, ...ENDWHILE...ITERATE...UNTIL and two forms of the case structure. All BASIC statements also be used. Over a dozen commands are included in the system. The translation routine not only produces a BASIC version of the program but also lists the structured program with automatic indention.

Copies: **Just released**
Price: **$35.00** (Texas residents add 5% sales tax)
Includes: System,Sample program on diskette. Documentation.
Author: **Robert F. Zant**
Available: Decision Systems
P.O. Box I3006
Denton, TX 76203

Name: **L.I.S.A.**
System: **Apple II**
Memory: **V1.5C: 48k**
V2.0: 64k
Language: **Machine**
Hardware: **Apple II, Disk II, Language card (V2.0) other options also**

Description: Lazer Systems' Interactive Symbolic assembler is interactive assembler. Syntax checking is performed at edit time resulting in immediate feedback for all syntax and addressing mode errors. Incorporates several special features such as the ability to store data in inverted or blinking mode, built-in disk operations, etc. Tokenizes the textfile so that less memory is required to hold a textfile in memory. V1.5C has room for 2,000 lines, V2.0 has room for 3,000. More can be given from the disk drive.

Copies: **Just released**
Price: **V1.5C $34.95**
V2.0 $49.95
Includes: Diskette w/software, documentation manual (100pgs)
Author: **Randall Hyde/Lazer Systems**
Available: Programma International, Inc.
3400 Wilshire Blvd
L.A., CA 90010

Name: **Interactive Statistics**
System: **Apple II Plus, or Apple II with Applesoft Firmware Board**
Memory: **48K RAM**
Language: **Apple Integer BASIC**
Hardware: **TV set and RF Modulator or Video Monitor, Apple disk drive, Optional printer and interface board**

Description: An analysis package designed for teachers and businessmen who do not have access to a large computer. The entire system is interactive, and its features include menu prompting and data stored in user-named variables. Available statistical routines range from the simple to the complex. Results can be displayed or printed in either columnar format, or plotted as histograms or simple X-Y graphs.

Price: **$149.00**
Available: Serendipity Systems
225 Elmira Road
Ithaca, N.Y. 14850

Name: **Advanced Mathematical Routines**
System: **Apple II plus or Apple II w/Applesoft Firmware Board.**
Memory: **48K RAM**
Language: **Apple Integer Basic**
Hardware: **TV set and RF modulator or Video monitor**

Description: Designed for use by professionals in the areas of research, business, and operations management. It consists of a set of mathematical tools which provide answers to a variety of common complex numerical problems in relatively short periods of time. The package includes routines for linear regression, matrix operations, numerical calculus, differential equations, and optimization. In addition, a routine is provided for the plotting of equations. Data sets, which are automatically stored on the diskette, can be recalled on demand.

Price: **$149.00**
Available: Serendipity Systems
225 Elmira Road
Ithaca, N.Y. 14850

Name: **Bowling Alley**
System: **OSI C-1P**
Superboard II
Memory: **4K up**
Language: **Basic**
Hardware: **Standard**

Description: Full graphics bowling game. Draws land and score sheet. Fast action and fun. Have many other programs for sale or trade. Send list of your programs.

Copies: **Just released**
Price: **$7.95**
Author: **Miles Hufford**
Available: M. Hufford
6715 E. Doubletree
Scottsdale, Arizona
85253

Name: **Musical Computer One and Two**
System: **Apple II**
Memory: **32K**
Language: **Integer Basic**

Description: A two-program cassette tape which explains the fundamental of music — including musical symbols and language, note reading on both the treble and bass clefs, telling time, note values and rests, piano keyboard, dynamic and tempo markings, signs and symbols and enough PRACTICE and TESTING opportunities for both the beginning and advanced student. Written by a M.A. educator with over 20 years of music experience. This is truly an alternative to music education, accompanied with colorful musical descriptions and musical sounds.

Copies: **Many**
Price: **$34.95, $1.00 s&h** (MI residents add 4 percent sales tax.)
Author: **Myra Marshall**
Available: Computer Applications Tomorrow
P.O.Box 605
Birmingham, MI 48012

6502 Bibliography: Part XXIII

Continuing bibliography of 6502 related material

Dr.William R. Dial
438 Roslyn Avenue
Akron, OH 44320

668. The Seed 1, No. 5 (October 1979)

Brown, Bob, "Sines and Cosines," pg. 11.
Program illustrating the use of the sine/cosine tables in the Apple.

Pope, Gene, "Tape Recorder Monitoring," pg. 12.
Simple hardware mod for your recorder enables you to monitor Apple programs as they are read in or out.

669. The Seed 1, No. 6 (Novemeber 1979)

Foens, Bob, "Applesoft Music/Tone Subroutine," pg. 5.
Apple tone routine.

McClure, Alan, "Write Protect Defeat," pg. 12.
A mod for the Apple Disk II.

Baga, Dennis, "Algol II Language for the Apple II Computer," pg. 13.
A translator for the Apple.

Reese, Chuck, "Analytical Apple," pg. 16.
Derivatives and Integrals on the Apple.

670. The Seed 2, No. 1 (January 1980)

Foens, Bob, "Deck of Cards," pg. 5.
How to select non-recurring random numbers.

Wagner, Roger, "Exceeding the Speed of Light with your Apple II," pg. 8-9.
Speed up your programs.

McClure, Alan R., "Reset Fix that Doesn't Void your Warranty," pg. 14.
Simple, easy to install hardware modification.

671. Interface Age 5, No. 4 (April 1980)

Roybal, Phil, "Data Base Management for the Apple II," pg. 94-101.
Description of File Cabinet.

Fox, Tom, "System of the Month - OSI C3-C," pg. 102-103.
A review.

672. Southeastern Software Newsletter, Issue 16 (Feb. 1980)

McClellan, George, "Peeks, Pokes, and Calls," pg. 1-4.
A tutorial on machine language. Apple.

Carpenter, Chuck, "The Whys of HEX Dump," pg. 4-7.
Learn all about memory dumps on the Apple.

Hartley, Tim, "Customize your Diskettes," pg. 7-8.
A Novel Hello proqram for the Apple Disk.
Short Applesoft program with graphics.

Anon., "Cube," pg. 9.
Pascal Turtlegraphics program.

Gustafson, Ralph D., "Machine Language Subroutine," pg. 10.
Subroutine to fetch the Input Output Block for the use of the Machine Language Programmer.

673. The Apple Shoppe 1, No. 5 (January 1980)

Smith, Brian, "Fun With Pascal," pg. 7-8.
A short program that illustrates some basics of programming in Pascal.

Anon., "Apple Assembly, Part 1," pg. 9-13.
A tutorial for the Apple Assembly program.

Crouch, Bill, "Down to Business," pg. 14-16.
Some routines useful in business applications of the Apple.

Anon., "Graphics in Assembly," pg. 17-20.
How to draw graphics from machine language programs.

674. The Paper 3, Issue 1 (February 1980)

Hennig, Doug, "Speeding Up the Print Statement," pg. 7-9.
Speed up your PET by a factor of four.

Lee, Arnie, "Floating Point Numbers," pg. 7-9.
All about Floating point and fractional numbers on the PET.

Sung, Wayne, "Himondis on a 16k PET," pg. 13-15.
A utility for the PET.

Kalmbach, Chris, "Modifications for PIMS" pg. 18.
Modifications for the PET information management program.

Sung, Wayne, "HIMONDIS and ROM Test Revisited," pg. 19-21.
Modifications for the PET dissembler and the ROM memory test.

Turco, Fran, "Better List," pg. 23-26.
An improved LIST routine for the PET.

675. Fort Worth Area Apple User Group 1, No. 7 (March 1980)

Meador, Lee, "Dissassembly of DOS 3.2," pg. 1-13.
Another installment of the Editor's comprehensive series, Apple.

Hoggatt, Ken, "Monitor commands in BASIC," pg. 14.
Enter monitor commands from Basic on your Apple.

Keyes, Patricia, "A String$ Expression Compiler for Applesoft," pg. 14.
An easier way to use data tables. Apple.

Crossman, Craig, "Fast DOS," pg. 15.
Init your Apple Disks in half the regular time.

676. The Seed 2, No. 2 (February 1980)

Davis, William, "Counting Files," pg. 5.
Count the number of Disk files on the Apple II disk.

Foens, Bob, "GEEJO," pg. 11.
Several short routines for the Apple.

Wagner, Roger, "Speeding in Applesoft," pg. 13.
Speed up program execution by bypassing REM statements.

Patton, J.W., "Applesoft/Prog. Aid #1 Music," pg. 14.
Modification of a program to use the Programmers Aid in Music on the Apple.

677. Stems From Apple 3, Issue 2 (February 1980)

Hoggatt, Ken, "Ken's Korner," pg. 3,7.
Similarities between Basic and Pascal.

Pfeiffer, Jim, "How Applesoft Stores String Arrays and A Tip on How to Clear String Array Space," pg. 4-6, 9.

Byerly, Kent, "Literal Input Fix," pg. 7.
An improvement on the Literal Input for Apple.

678. Abacus Newsletter 2, Issue 2 (February 1980)

Anon., "Hi-Res Flag and Music," pg. 4.
A graphics program for the Apple II.

Crossman, Craig, "Colossus," pg. 6-7.
Listing for a new game for the Apple.

Crossman, Craig, "Apple Tricks," pg. 7.
How to make your Apple programs unlistable.

Anon., "Floppy Disk," pg. 8.
All about Floppy Disks, hard vs. soft sector, etc.

Crossman, Craig, "Fast DOS," pg. 9.
Speed up your Apple DOS.

Anon., "Introduction to Simulation," pg. 10-14.
A good tutorial on Simulation for the Apple.

Wright, Don, "Disk II Utility: Disk Map Display," pg. 15-17.
An interesting disk utility for the Apple Disk System.

Brown, A.R.,Jr., "Patriotic Display," pg. 18-19.
A graphics and music program for the Apple.

Schwartz, Dana J., "Random Spiro," pg. 19.
A short program for complex Hi-Res graphics on the Apple.

679. MICRO No. 22 (March 1980)

Pruetz, Harry L., "Apple II Floating Point Utility Routines," pg. 7-11.
A guide to Applesoft Basic floating Point routines.

Finn, Kenneth, "A Machine Language Screen Print Program for the Old or New PET," pg. 13-14.
Program which gives user control over the PET screen.

Carlson, Edward H., "Polling OSI's Keyboard," pg. 17-18.
The Polled Keyboard technique permits the user to define the function of various keys to his own specifications.

Kershner, Carl J., "A Digital Thermometer for the Apple II," pg. 21-23.
Use thermistor probes connected directly to the Apple II Game I/O Connector and convert to Celsius or Fahrenheit.

Lary, Richard A., "Challenger II Cassette Techniques," pg. 25-26.
Storage and retrieval of sequential data files on cassette.

DeJong, Dr. Marvin, "Beginning Boolean: A Brief Introduction to Boolean Algebra for Computerists," pg. 29-33.
A tutorial on computer math and logic functions.

Vrtis, Nicholas, "Program Checksum Calculator," pg. 39-40.
SYM-1 checksum program.

Tripp, Robert M., "Ask the Doctor," pg. 42-43.
A technique to solve a problem in the AIM TTY service, and a program for easy tape retrieval on the KIM.

Guest, Ronald A., "Clocking KIM," pg. 45-48.
A tutorial involving techniques for a KIM clock program.

McLean, William, "A Home Message Center," pg. 53-55.
Use your Apple as a bulletin board for the family.

Bullard, Gary J., "Stop That PET!," pg. 57-61.
How to stop the run-away program without hitting RESET.

Rowe, Mike (Staff), "The MICRO Software Catalogue: XVIII," pg. 63-65.
MICRO lists 14 new 6502 programs.

Dial, Dr. William R., "6502 Bibliography, Part XVIII," pg. 67-70.
About 150 more references to the 6502 literature.

680. Call — Apple 3, No. 2 (February 1980)

Golding, Val J., "Lower Case and Graphics Printer Routine," pg. 7-23.
Apple program for printing using the Integral Data printer.

Cadmus, Ray, "Formatted Print," pg. 25.
Formatted print on the Apple for money programs.

Capes, Nelson R., "Apple II Disassembler," pg. 31.
Improved disassembler for the Apple.

Vaughan, Craig, "Program to Modify the BIOS Modules," pg. 32-33.
Modification for the Apple to remedy the inability to display lower case from the Pascal system.

Wagner, Roger, "APPENDectomies in Integer and Applesoft," pg. 34.
How to remove Appended Binary data from basic programs.

Golding, Val J., "Routine to Print NBumbers above 32767 in Integer Basic," pg. 34.
Special routine for the Apple.

Winston, Alan B., "The Multi-Lingual Apple," pg. 35-36.
Notes on Pascal for the Apple.

Vaughan, Claudia M., "Bridging the Gap," pg. 36-37.
A tutorial to help you over the rough spots in Pascal.

Williams, Rick, "Random Drill Tutorial," pg. 38-40.
Enter your own question/answer pairs in this drill program.

Foote, Gary and Barnes, Keith, "File Cabinet Improvements, Ad Infinitum," pg. 41-42.
More modifications for this popular program.

Hyde, Randall, "The Apple Doctor," pg. 43.
More on Pascal including printer output and turnkey operation.

Huelsdonk, Bob, "Formatting Re-Visited," pg. 48.
A 'print using' routine for the Apple.

Golding, Val J., "Crawler," pg. 52.
A short routine to print a message which crawls across the Apple screen.

681. Rubber Apple Newsletter (September 1979)

Throop, Gilbert D., "Rounding Off," pg. 2.
Short routine for rounding off numbers on the Apple.

Wysocki, Tom, "Billboard Display Routine," pg. 2.
Move your message across the Apple screen.

Gabelman, Ken, "Sort-File," pg. 3.
A sort routine for the Apple.

682. OSI User Independent Newsletter, No. 2 (March 1980)

Curley, Charles, "OS-65U Notes," pg. 1-2.
Notes on the use of the COPYFI and LABLE programs.

Curley, Charles, "OS-65D Notes," pg. 2-3.
How to use dual disk drives to load games for visitors.

683. Stems From Apple 3, Issue 3 (March 1980)

Hoggatt, Ken, "Ken's Korner," pg. 3.
More about the similarities of Applesoft and Pascal,

giving examples of programs in each language (String Demo).

Sittel, Randy, "Hobby Hardware," pg. 5-6.
Using game output ports (Part I) — Light LEDs with the Apple.

Anon., "Applications Note for Pascal," pg. 7-8.
A fix for the LONG INTEGERS in the Apple Pascal language.

Hoggatt, Ken, "Statistics for Throwing Three Dice," pg. 9.

Hoggatt, Ken, "Read Names," pg. 13.
A sort and search routine for Names or lists on a data statement.

684. Kilobaud Microcomputing, No. 39 (March 1980)

Baker, Robert W., "PET Pourri," pg. 7-9.
Discussion of the BASIC SWITCH, Digitizer tablet, a Date Book program, etc.

Anon., "OSI Small Systems Journal," pg. 10-13.
The first of a two part series on multiple user systems.

Flippin, Allan, "Assembly-Language Benchmarks," pg. 26-35.
Compared against a number of 8-bit microprocessors as well as 16-bit systems and the IBM 370-145, the 6502 system was second only to the 370 in execution speed.

Derfler, Frank J., "Dial-Up Directory," pg. 50-51.
All about the use of Modems and Bulletin Boards and shared use of large data base systems.

Rowe, John and Grossman, Chris, "Multiple Page Graphics for the Apple II," pg. 66.
A program to flip through the pages of your Apple.

Wadsworth, Nat, "It's Time to Draw the Line," pg. 78-80.
Here's how to draw a straight line with the PET.

Paturzo, Bonaventura, "Stand-Alone Video Terminal," pg. 94-97.
A SYM-1 microcomputer system using a large-scale integration CRT controller.

Bramblet, Timothy L, "Keyed-Up PET," pg. 132-133.
Add a keyboard to your PET for less than $60.00.

Buszta, Ron, "An Apple a Day Keeps a Kid Occupied," pg. 172.
Experiences with a five year old and a computer.

Putney, C.B., "Stacking Program," pg. 184-185.
How to stack multiple games or programs in one Basic load.

Gordon, Hal T., "Instruction Sets Examined and Compared," pg. 192-194.
Comparing the 6502, 8080, Z-80, 6800 and 2650 microprocessors.

Wayne, Phil, "Pascal Workshop," pg. 20-22.
How to patch the Pascal BIOS to change your Apple I/O system.

Chavez, Franklin, J., "Eight Queens Problem, Apple II Style," pg. 23-26.
A Pascal solution to the Eight Queens Program.

685. Rubber Apple Newsletter (November 1979)

Musgrave, J. Scotty, "Apple Monitor," pg. 2.
A tutorial on the Apple Monitor.

686. Rubber Apple Newsletter (January 1980)

Musgrave, J. E., "DOS Identifier," pg. I
Simple routine for identifying the DOS in use.

687. Compute No. 3 (March/April., 1980)

Stuart, Chuck, "A Preview of Commodore's New Disk Basic 4.0," pg. 10-11.
Describes 14 new disk commands.

Isaacs, Larry and Compute Staff, "Enhancing Commodore's Word Pro II," pg. 15-16.
Upgrading the Word Pro II program for a Commodore Business Machine system.

Wadleigh, Hal, "File Conversion on the Commodore 2040 Drive," pg. 18-21.
Program listing for a conversion program.

Bruey, Alfred J., "Using the GET Statement on the PET," pg. 23-24.
How to use the GET statement instead of INPUT to enter data.

Butterfield, Jim, "Machine Language,"
General discussion of machine language for the PET/CBM systems.

Isaacs, Larry, "UTINSEL: Enabling Utilities," pg. 34-37.
Program, with listing, to provide a flexible solution to several PET problems involving CHARGOT, WEDGE, etc.

Lindsay, Len, "Identify Your Atari Colors," pg. 39.
A short program to identify on the ATARI.

Semancik, Susan, "Manual Alphabet Tutorial on a PET," pg. 4148.
Use the PET to help the handicapped. Drawing hands for sign language.

Pratto, Marlene, "The Learning Lab," pg. 51-53.
This program will help teachers calculate term and partial term grades, means, deviations, etc.

Heise, David R., "Light-Pen Selection from Large Menus on the PET/CBM Screen," pg. 60-62.
A light-pen can select from menus of over 90 items.

Kemp, D.P., "Naming Apple Cassette Files," pg. 68-69.
A short program to support Named files on Apple cassettes.

Wiplich, Mike, "On Interfacing an Apple II to A Heathkit H-14," pg. 69.
Software compatible with the Apple DOS is given for using the Heath printer.

Lindsay, Len, "ATARI Program Saving," pg. 76.
Improve your tape SAVE/READ routine.

Bunker, W.M., "Lower Case Descention," pg. 81.
Improve the lower case program for your PET.

Wuchter, Earl H., "Comments on Null Return (LINPUT) Simulation," pg. 81.
Improvements for this PET Program.

McCarthy, C.A., "Cheep Print--Hard Copy for Soft Cost," pg. 82-86.
Use your Audio Cassette as a substitute for hard copy using the PET.

Lindsay, Len, "Direct Screen Input," pg. 88.
Use your PET Video screen as input into your programs.

Herman, Harvey B., and Pate, Charles B., "A Versatile Serial Printer Interface for the PET," pg. 90-91.
A simple design for a serial interface.

Strasma, Jim, "MAE, a PET Disk-Based Macro Assembler," pg. 93.
This assembler, compatible with CBM and other printers offers many unique features.

Strasma, Jim, "The PET Rabbit," pg. 94.
The PET Rabbit is an extension of the PET operating system.

MICROScope Outlined

MICRO publishes its first MICROScope review. With its appearance we would like to point out the seriousness of our review intentions. Shown below is a copy of the Reviewer's Responsibilities involved when a review is written. Manufacturers will be provided with sufficient time to read the review before it is published.

If you are a manufacturer with a product you would like to submit for impartial review, please contact us and we shall send you all our MICROScope information and necessary forms.

The Reviewer's Responsibilities

In agreeing to evaluate a product and write a MICROScope product review, you accept a definite set of responsibilities. It is important that you seriously consider the impact of this undertaking. If at any time, for any reason, you feel that you may not be able to completely handle the responsibilities as outlined below and in the **MICROScope Product Reviewer Agreement**, please immediately contact the MICRO staff and resign the particular reviewing assignment. This will not be considered a failure on your part, but rather will reflect well on your sincerity and honesty. It will in no way jeopordize or disqualify you from other review assignments.

Responsibility to the Reader. Due to the method by which unbiased reviews are being generated for the **MICROScope** section of MICRO, readers will be placing great faith in the accuracy, validity, completeness, and fairness of the reviews. Many readers may base their decisions to purchase, or not to purchase, an item on the review which you give it. They are counting on the reviewer to provide them with all of the detailed information about the product, its good points and its bad, so that they can make an informed judgment. If you submit less than your absolute best review effort, then you will have failed in this responsibility.

Responsibility to the Manufacturer. The manufacturer has submitted his product for review in order to obtain the publication of an impartial review which will, hopefully, result in increased sales of his product. He has the right to expect that all of the product's features will be fully explored, fully understood, and adequately described in the review. He also has the right to expect that all information which is essential for a potential purchaser will be included. He has a legal right to expect that the review will be unbiased and that the reviewer does not have any conflict of interest at any level that could effect his judgment and his review.

Responsibility to MICRO. MICRO has not published the new releases and other manufacturer generated promotional material which abounds in other magazines, since we felt that this material was, by its very nature, biased. The method we are using to keep our readers informed, the **MICROScope** reviews, is costing us time, money, and a lot of effort. In addition, MICRO is putting its reputation on the line. Since we are sponsoring these reviews, any review which is less than perfectly honest reflects poorly on us.

This brief discussion of responsibilities is not intended to scare any legitimate reviewer away from the task. It is intended to make every reviewer consider his obligations and make certain that he feels that he can perform the reviewing task without reservation.

This is how we expect the **MICROScope** review process to work. With our great reviewers, we know we will receive the best results.

Missing MICRO Information?

MICRO is devoted exclusively to the 6502. In addition, it is aimed at useful, reference type material, not just "fun and games". Each month MICRO publishes application notes, hardware and software tutorials, a continuing bibliography, software catalog, and so forth. Since MICRO contains lots of reference material and many useful program, most readers want to get the entire collection of MICRO. Since MICRO grew very rapidly, it quickly became impractical to reprint back issues for new subscribers. In order to make the older material available, collections of the reprints have been published.

[A limited number of back issues are still available from number 7 to 18 and 20 to current. There are no 19's left.]

The BEST of MICRO Volume 1 contains all of the significant material from the first six issues of MICRO, from October/November 1977 through August/September 1978. This book form is 176 pages long, plus five removeable reference cards. The material is organized by microcomputer and almost every article is included. Only the ads and a few 'dated' articles have been omitted. [Now in third printing!]

Surface...$7.00 **Air Mail...$10.00**

The BEST of MICRO Volume 2 covers the second six issues, from October/November 1978 through May 1979. Organized by microcomputer, this volume is 224 pages long.

Surface...$9.00 **Air Mail...$13.00**

The BEST of MICRO Volume 3, covering the twelve issues from June 1979 through May 1980, will be over 400 pages long. It is scheduled for late summer 1980. The price is still to be determined.

For a free copy of the Index for Volumes 1, 2, and 3, please send a self-addressed, stamped envelope to:

BEST of MICRO, P.O. Box 6502, Chelmsford, MA 01824

Advertisers' Index

ACTION, STRATEGY, AND FANTASY for the SERIOUS games player and his APPLE II

Brain Games - 1 demands ingenuity.

Two players bombard radioactive material with protons and electrons until it reaches critical mass and sets up a ***Nuclear Reaction. Dodgem*** *requires you to outmaneuver another player to get your pieces across the board first.* ***Dueling Digits*** *and* ***Parrot*** *challenges your ability to replicate number and letter sequences.* ***Tones*** *lets you make music with your Apple (16K) CS-4004 $7.95.* ***Strategy Games*** *and* ***Brain Games*** *are on one disk (16K) CS-4503 $14.95.*

THINK

Strategy Games - 1 keeps games players in suspense.

You and your opponent trail around the screen at a quickening pace attempting to trap each other in your ***Blockade.*** *A 7 category quiz game will certify you as a* ***Genius*** *(or an errant knave!). Beginners will meet their master in* ***Checkers. Skunk*** *and* ***UFO*** *complete this classic collection (16K) CS-4003 $7.95*

Know Yourself through these valid self-tests.

Find out how your life style effects your ***Life Expectancy*** *or explore the effects of* ***Alcohol*** *on your behavior.* ***Sex Role*** *helps you to examine your behavior and attitudes in light of society's concept of sex roles.* ***Psychotherapy*** *compares your feelings, actions, and phobias to the population's norms and* ***Computer Literacy*** *tests your microcomputer savvy. A fun and instructional package (16K) CS-4301 $7.95.* ***Know Yourself*** *and* ***CAI Programs*** *are on one disk (16K) CS-4503 for $14.95*

IMAGINE

You're in command in *Space Games* - 1.

Maneuver the TIE fighters into your blaster sights and zap them with your lasers to save the rebel base camp from annihilation in ***Star Wars. Rocket Pilot*** *is an advanced real time take off and landing game. High resolution graphics, exploding saucers and sound effects add to the suspense as you repel the* ***Saucer Invasion.*** *Finally, a bonus graphics demonstration,* ***Dynamic Bouncer*** *(16K) CS-4001 $7.95.* ***Space Games*** *and* ***Sports Games*** *are on one disk (16K) CS-4501 for $14.95*

ACTION

Sports Games - 1 puts you in the Apple World Series

Take the field in the ***Great American Computer Game.*** *Mix up your pitches to keep the batter off balance. Move your fielders to snag the ball before he gets to first. Balls and strikes, double plays, force outs, and errors let you play with a realistic strategy. Also in the line up—* ***Slalom,*** *a championship downhill ski race,* ***Torpedo Alley,*** *and* ***Darts*** *(16K) CS-4002 $7.95.* ***Space Games*** *and* ***Sports Games*** *are on one disk (16K) CS-4501 for $14.95*

It's easy to order *SENSATIONAL SOFTWARE* for your Apple II.

Send payment plus $1.00 shipping and handling in the U.S. ($2.00 foreign) to Creative Computing Software, P.O. Box 789-M, Morristown, N.J. 07960. N.J. residents add $1.00 sales tax. Visa, Master Charge and American Express orders may be called in toll free to **800-631-8112** (in N.J. 201-540-0445).