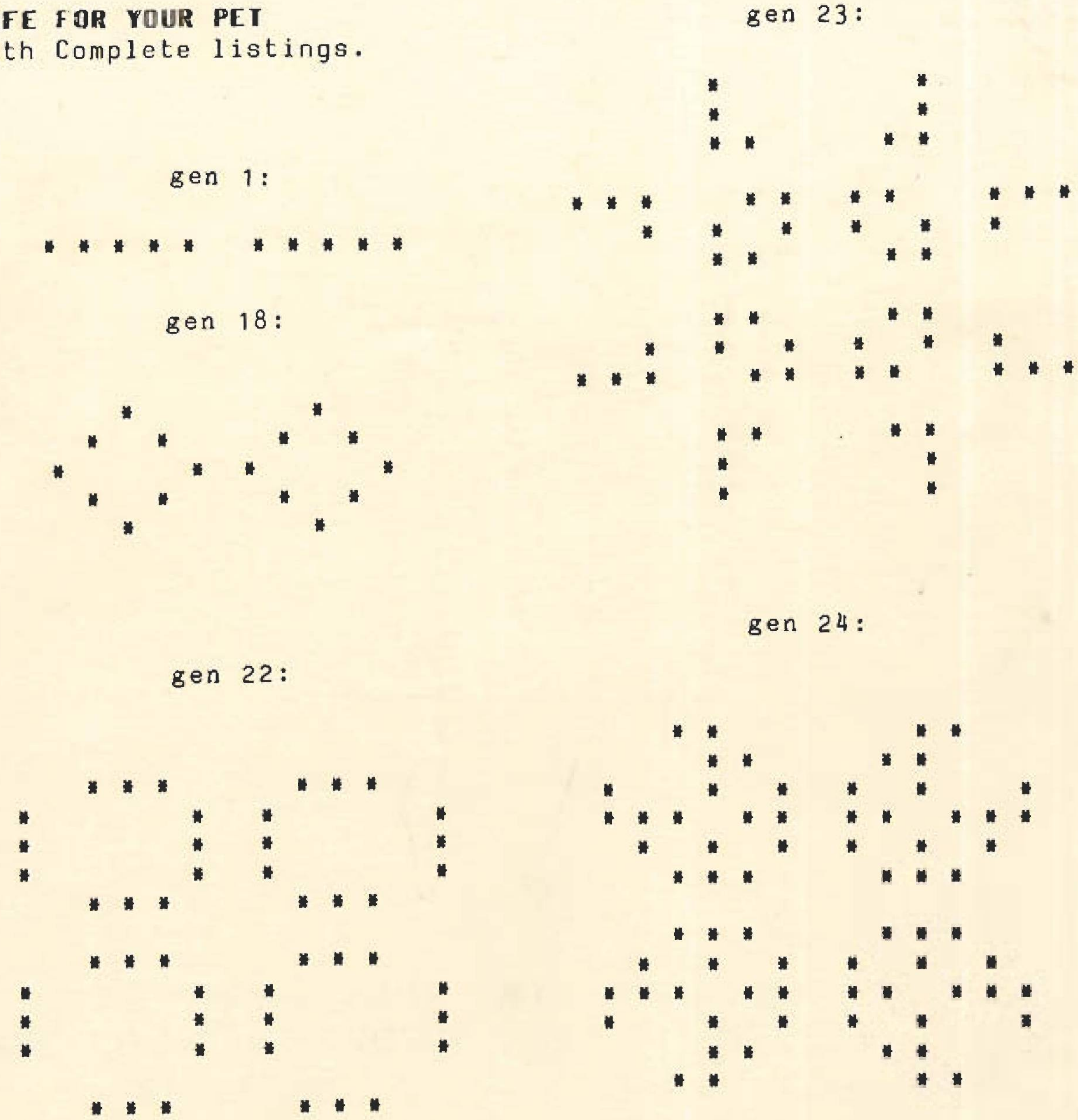
MICRO

THE 6502 JOURNAL

LIFE FOR YOUR PET
With Complete listings.



RMT

NO 5 **June - July 1978** **$1.50**

WE'RE THE APPLE EXPERTS

Check our low prices and large selection of computers, software and peripherals.

APPLE II PERIPHERAL INTERFACE CARDS:

■ FLOPPY DISC SYSTEM ($2,300)

—Programs Saved and Loaded by Name
—Powerful Firmware DOS File Handling Capability
—252K Bytes Storage Capacity, 8 Inch Dia. Disc
—Capable of Utilizing Up to 4 Drives (One Million Bytes)
—File Handling as Easy as Inputing or Printing
—Access Methods: Stream, Punctuated, Relative, Direct
—Dual Disc ($2,800)

■ PROGRAMMABLE PRINTER INTERFACE ($80.00)

—Onboard EPROM Printer Driver
—Full Handshake Logic
—High Speed Parallel Output Port Capability
—Provision for 256 Byte I/O Drive in EPROM
—Printer, Driver Programs Available for Centronic, SWTPC-40, and Other Printers

■ PROTOTYPING BOARD, 2¾"x 7" ($24.95)

■ APPLE POWER CONTROL INTERFACE

—This interface plugs into any peripheral slot on the Apple II board and provides 16 channels of control. Power Control modules plug into the interface via a ribbon cable. Each Power Control module provides 4 seperate 110V A.C. Circuits at 12 amps. Up to 4 Power Control Modules may be used with each interface.
—Control Room Lights, Stereo Equipment, Security Systems, Electrical Appliances
—Handle Up to 1000 Watts per Channel Directly From Program Control
—Complete Isolation of the Computer From the AC Line
—PRICE—
- Apple Power Interface Board and One Power Control Module ($95.00)
- Additional Power Control Modules (Controls Four AC Circuits) ($35.00)

ADD ON MEMORY FOR APPLE II

■ Set of Eight	4K RAM CHIPS	$ 24.00
■ Set of Eight	16K RAM CHIPS	$240.00

BOOKS

■ 6502 Programming Manual for Apple II or PET	$ 7.95
■ 6502 Hardware Manual for Apple II or PET	$ 7.95

Program Code

Program Name: **I—For Integer Basic / A—For Applesoft / M—For Machine Language** / **Memory Size** / **S—For Sound / G—For Graphics / SG—For Both** **Price**

Games:

Super Othello	I/12K/G	$10.00
True Las Vegas Blackjack	I/16K/GS	$10.00
Qubic (3-D TIC TAC TOE)	I/8K/G	$ 5.00
Saucer Invasion	M/16K/GS	$10.00
Star Wars	M/16K/GS	$10.00
Rocket Pilot	M/16K/GS	$10.00

Education Programs:

Supermath (+, −, *, /)	I/8K/G	$ 7.50
Memory Aide	I/8K	$ 7.50
Study Aide	I/8K	$ 7.50
Don't Fall	I/8K	$ 7.50
True/False Quiz	I/8K	$ 7.50
Matching Quiz	I/8K	$ 7.50

Applications:

Co-Resident Assembler	M/16K	$20.00
Memory Test	M	$15.00
Apple Music (3 Octaves)	M/12K/S	$15.00
Floating Point Datasave	M/16K	$15.00

Graphics & Sound:

Drawing	I/8K/G	$ 7.50
Message Board	I/8K/G	$ 7.50
Keyboard Organ	I/8K/S	$ 7.50
Example Sounds	I/8K/S	$ 5.00
Kaliedescope Display	M/8K/G	$ 5.00
Color Eater Display	M/8K/G	$ 5.00
Life	M/16K/G	$ 5.00
B. Bishop's Dancing Man	M/16K/GS	$15.00

Business:

Word Processor (Lower Case)	I/20K	$30.00
Word Processor (Upper Case)	I/20K	$30.00
Word Processor (Upper and Lower Case)	A/16	$50.00
Business Inventory	I/20K	$50.00
Master File Maintenance	I/20K	$50.00
Point of Sales	I/20K	$50.00
Reorder Report	I/20K	$50.00
File Sort	M/20K	$20.00

**Computer Components
of Orange County
6791 Westminster Ave.
Westminster, Calif. 92708
(714) 898-8330**

**Orange County's only
AUTHORIZED COMMODORE
PET DEALER**

**Send for software and
hardware information.**

Mastercharge, Visa, B of A accepted. No C.O.D. Allow two weeks for personal check to clear. Add $1.50 for handling and postage. For computer system, please add $10.00 for shipping, handling, and insurance. California residents add 6% sales tax.

MICRO

JUNE/JULY 1978

ISSUE NUMBER FIVE

Advertisers Index

Computer Components	IFC	New England Electronics Co.	27
Computer Shop	2	Speakeasy Software	31
Riverside Electronics	8	The Computer Store	IBC
The COMPUTERIST, Inc.	20	The Enclosures Group	BC

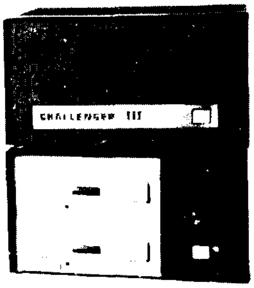
MICRO is published bi-monthly by The COMPUTERIST, Inc., 56 Central Square, Chelmsford, MA 01824. Robert M. Tripp, Editor/Publisher. Controlled Circulation postage paid at Chelmsford, Massachusetts.

Single Copy: $1.50 Annual Subscription: $6.00 (6 issues) in USA.

COMPUTER SHOP

288 NORFOLK ST. CAMBRIDGE, MASS. 02139
corner of Hampshire & Norfolk St. 617-661-2670

NOW WE HAVE OSI AND YOU CAN GET ONTO THE BUS

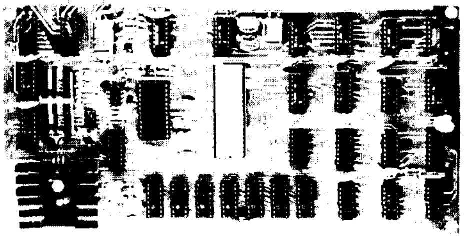


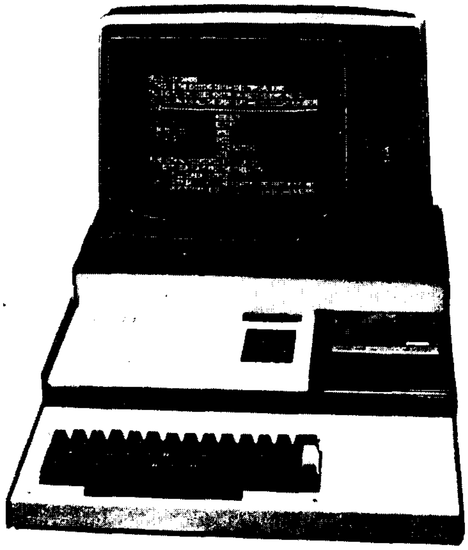
OHIO SCIENTIFIC

We carry the entire O. S. I. line of equipment. at the time of writing this ad, the items marked with * were in stock:

500 Computer on a board with 8K ROM Basic-Serial-	$298.00
500-1 as above but with Power Supp-Cabinet-	$498.00
C2-4P As above with keyboard and 32x64 Vid. & Cass...	$598.00*
C2-8P Same as 4P but 8 slot MB and Big Power Supply...	$825.00*
C2S1S Single Disk System 16 K Mem. Serial	$1990.00*
C3-S1 Dual Disk Triple Processor 32 K Memory	$3590.00*

The CS100 VIDEO TERMINAL BOARD IS A 16 LINE BY 64 CHARACTER DISPLAY GENERATOR WITH CURSOR CONTROL AND EDITING Connect a 5V. ASCII Keyboard to it, a Regulated 5 Volt, Unregulat- 8 Volts, or 8 Volts AC, and your KIM Teletype port to it along with a video monitor and away you go with all the convenience of a Video Terminal on your KIM.





KIMS AND UPGRADES

CM-1 4K 1MHz. Memory	125.00
CM-2 4K 2MHz. Memory	149.00
CM-3 16K 1.5 MHz. Ultra Low Pwr. Mem...	596.00
CA-7 I/O Board	399.00
480 Backplane..Motherboard	39.00
all above are OSI products available from us.	
VF8 4K Memory assembled & tested.	129.00
for low power RAM add	10.00
same in kit form	74.50
full set of sockets for Kit	10.00
VF8 Motherboard buffered for 4 Boards	65.00
Connector Assembly for KIM to VF8	20.00
8K S100 Memory Board with instructions.K	165.00
same but fully assembled and tested	199.00
CS100 Cabinet cut out for KIM	129.00
3 Connector S100 Motherboard Assembly	75.00
CGRS S100 TIM Kit	129.00
CGRS S100 6502 CPU Kit	179.00
CGRS S100 Front Panel Kit	129.00
XITEX Video Terminal Board 16X64.K	155.00
XITEX Video Terminal Board Assembled...	185.00
KIM-1	245.00
CS100 with CGRS, Xitex, 16KRAM, TV, KB	1529.00
Same but Assembled	1989.00
PS-5 Pwr Supp. 5V5A9V1A-12V1A6x6X2	75.00
PS-5 Assembled	90.00

Total of Order..Circle Items wanted.$..............
Mass. Residents Sales Tax 5%........$..............
Shipping, 1%($2.00 min.)................$..............
Total Remittance or Charge............$..............

BAC, VISA, MC NO. ....

SIGNATURE..

NAME..

ADDRESS..

CITY.......................STATE.........ZIP......

IN THIS ISSUE . . .

It's always nice to be able to have fun while learning. "Life for your PET" by Dr. Frank H. Covitz presents the amazing game of Life, implemented on a PET. This remarkable game, which was the subject of a number of Martin Gardner Scientific American columns, uses a few simple rules to generate a very complex universe. It is ideally suited to a microcomputer with a display. The program presented here is written in 6502 assembly code, not BASIC, and this will be illuminating in itself to many PET owners. In addition, it demonstrates how to use the PET display directly.

While the PET people can be playing Life on their machines, the Apple folk can be playing music on theirs, thanks to the "Applayer Music Interpreter" of Richard F. Suitor. A couple of songs are included, but most users will want to generate their own following the techniques described. The complete source listings also should help novice programmers understand the 6502 better.

One thing that the above two articles have in common is their use of 6502 assembly level code. Since many users do not have assemblers, and will therefore be keying the code into their machine by hand, it would be nice to have a disassembler which converted the code in the computer back into a readable form. "A BASIC 6502 Disassembler for Apple and PET" by Michael J. McCann can do the job. Written entirely in BASIC, it will disassemble code on a PET or Apple, using the MICRO 6502 Syntax. In addition to its obvious utility value, the program is particularly instructive in its handling of alphabetic strings.

KIM-1 owners will find "A Block Hex Dump and Character Map Utility Program for the KIM-1" by J. C. Williams to present a neat utility for dumping to a terminal. While the KIM-1 Monitor has a built-in Dump, it's format leaves a lot to be desired. This utility has a more useable format, plus it provides the option of having data printed as alphabetic characters as well as hex.

When listing to a hardcopy device, the faster the printing the better. Not so when going to a display. For a display you would like to have some way to slow down the display, stop it when you get to a particular portion, and then continue or abort the listing. Well, if you are an Apple owner, you are in luck because Bob Sander-Cederlof has provided "A Slow List for Apple BASIC". The program is written in 6502 assembly language and presents some insights into the workings of the Apple Monitor.

We are fortunate to have, starting in this issue, a series of tutorial articles by Marvin L. De Jong on "6502 Interfacing for Beginners". Marvin has already contributed a number of excellent articles to MICRO, and this series sounds like exactly what many readers have specifically requested. This months installment covers "Address Decoding". In addition to "talking at you", the article provides a number of experiments you can perform to really understand what is happening.

William Dial's "6502 Bibliography" continues with part IV. Since so much is being written about the 6502, finally, we are having to restrict the coverage somewhat. From now on, references to obscure journals, new product notes and ads, minor letters or notes or corrections, etc. will not be included. Also, references to the KIM-1 User Notes will be combined and brief since it is assumed that most MICRO readers already get KUN (if not, they should).

A few new products are presented:

"Rockwell's New R6500/1" is a new chip that looks very interesting for many of those applications which need processing power but not a lot of memory or fancy features. The R6500/1 combines a 6502 with 2K bytes of ROM, 64 bytes of RAM, 32 programmable I/O lines, timer, and a few other features, all in a single 40-pin package.

"Synertek's VIM-1" is a new 6502-based system which is an upgrade of the KIM, designed as an easily expandable system with many of the KIM-1 features, plus a number of new wrinkles. The single piece price is $270 and is scheduled for delivery soon.

"Rockwell's AIM is Pretty Good" discusses an exciting new single-board microcomputer which features a full ASCII keyboard, 20 character display and a 20 character printer, for $375!

NOTES, ANNOUNCEMENTS, ETC.

The NOTES

Henry Ball of Burbank CA notes that: "The K7 connection on KIM provides a convenient control for the motor on a cassette tape player/recorder. Just connect a relay circuit to it and, without any further programming, it will obediently start and stop the recorder for the 1873 READ and any Supertape routine. Tryit, you'll like it."

Robert A. Huelsdonk of Seattle, WA, referring to the Apple Printer articles, suggests the following:

"Printer CALL Commands:

```
   Integer BASIC:
     ON:  CALL 896
     OFF:  PR#0
   Applesoft BASIC:
     ON:  X=USR (896)
     OFF:  POKE 54,240:POKE 55,253
```

These commands can be entered from the keyboard or in a program statement. If a printer other than a 40 column is used, then it is also necessary to POKE 33,40 to return the CRT to it's normal window width."

Robert M. Tripp of Chelmsford, MA notes that a number of people were mislead by the "Typesetting" article into thinking that he had a Diablo Hytype Printer hooked directly to his KIM-1. Actually the printer is part of a terminal which talks to the KIM via standard 20MA current loop methods. A reader from New Guinea has promised an article on how to directly hook up a Diablo, and says that it is easy.

The ANNOUNCEMENTS

The MICROCOMPUTER RESOURCE CENTER Inc. offers a number of services including a free publication devoted to the PET, the "PET GAZETTE". A PET Cassette Exchange is also being set up in which you submit one program and get two-to-four programs in return. For your free subscription or other info, write:

Len Lindsay, Editor
PET GAZETTE
1929 Northport Drive No. 6
Madison, WI 53704

6502 GROUPS

Interested in starting a KIM-1 Users Club in the San Fernando Valley Area

Jim Zuber
20224 Cohasset No. 16
Canoga Park, CA 91306
213/341-1610

THE APPLE CORE

Scot Kamins, Organizer
Box 4816 Main Post Office
San Francisco, CA 94101

THEATER COMPUTER USERS GROUP

A number of KIMs being used by members. Dues $4.00 include newsletter.

Mike Firth
104 N. St. Mary
Dallas, TX 75214

A.P.P.L.E.

Val J. Golding, President
6708 39th Avenue SW
Seattle, WA 98136
206/937-6588

MICRO 6502

New group forming in New England to pursue and support serious 6502 efforts

Robert M. Tripp, Organizer
P.O. Box 3
S. Chelmsford, MA 01824
617/256-3649 Days

*** Send us your club information ***
** Due to our publication schedule **
*** meeting announcements should ***
*** cover several months - Sept/Oct ***
*** for the Aug/Sept issue ***

The ETC.

AUTHORS

MICRO is currently paying $10/page for original articles. See "Writing for MICRO" 4:33 and the "Manuscript Cover Sheet" 4:34 for basic info. The deadline for any issue is about the end of the first week in the month prior to publication, e.g. July 10th for the August/September issue.

LIFE FOR YOUR PET

Dr. Frank H. Covitz
Deer Hill Road
Lebanon, NJ 08833

Since this is the first time I have attempted to set down a machine language program for the public eye, I will attempt to be as complete as practical without overdoing it.

The programs I will document here are concerned with the game of "LIFE", and are written in 6502 machine language specifically for the PET 2001 (8K version). The principles apply to any 6502 system with graphic display capability, and can be debugged (as I did) on non-graphic systems such as the KIM-1.

The first I heard of LIFE was in Martin Gardner's "Recreational Mathematics" section in Scientific American, Oct-Nov 1970; Feb. 1971. As I understand it, the game was invented by John H. Conway, an English mathematician. In brief, LIFE is a "cellular automation" scheme, where the arena is a rectangular grid (ideally of infinite size). Each square in the grid is either occupied or unoccupied with "seeds", the fate of which are governed by relatively simple rules, i.e. the "facts of LIFE". The rules are: 1. A seed survives to the next generation if and only if it has two or three neighbors (right, left, up, down, and the four diagonally adjacent cells) otherwise it dies of loneliness or overcrowding, as the case may be. 2. A seed is born in a vacant cell on the next generation if it has exactly 3 neighbors.

With these simple rules, a surprisingly rich game results. The original Scientific American article, and several subsequent articles reveal many curious and surprising initial patterns and results. I understand that there even has been formed a LIFE group, complete with newsletter, although I have not personally seen it.

The game can of course be played manually on a piece of graph paper, but it is slow and prone to mistakes, which have usually disasterous effects on the final results. It would seem to be the ideal thing to put to a microprocessor with bare-bones graphics, since the rules are so simple and there are essentially no arithmetic operations involved, except for keeping track of addresses and locating neighbors.

As you know, the PET-2001 has an excellent BASIC interpreter, but as yet very little documentation on machine language operation. My first stab was to write a BASIC program, using the entire PET display as the arena (more about boundaries later), and the filled circle graphic display character as the seed. This worked just fine, except for one thing - it took about 2-1/2 minutes for the interpreter to go through one generation! I suppose I shouldn't have been surprised since the program has to check eight neighboring cells to determine the fate of a particular cell, and do this 1000 times to complete the entire generation (40x25 characters for the PET display).

The program following is a 6502 version of LIFE written for the PET. It needs to be POKE'd into the PET memory, since I have yet to see or discover a machine language monitor for the PET. I did it with a simple BASIC program and many DATA statements (taking up much more of the program memory space than the actual machine language program!). A routine for assembling, and saving on tape machine language programs on the PET is sorely needed.

The program is accessed by the SYS command, and takes advantage of the display monitor (cursor control) for inserting seeds, and clearing the arena. Without a serious attempt at maximizing for speed, the program takes about 1/2 second to go through an entire generation, about 300 times faster than the BASIC equivalent! Enough said about the efficiency of machine language programming versus BASIC interpreters?

BASIC is great for number crunching, where you can quickly compose your program and have plenty of time to await the results.

The program may be broken down into manageable chunks by subroutining. There follows a brief description of the salient features of each section:

MAIN (hex 1900)

In a fit of overcaution (since this was the first time I attempted to write a PET machine language program) you will notice the series of pushes at the beginning and pulls at the end. I decided to save all the internal registers on the stack in page 1, and also included the CLD (clear decimal mode) just in case. Then follows a series of subroutine calls to do the LIFE generation and display transfers. The zero page location, TIMES, is a counter to permit several loops through LIFE before returning. As set up, TIMES is initialized to zero (hex location 1953) so that it will loop 256 times before jumping back. This of course can be changed either initially or while in BASIC via the POKE command. The return via the JMP BASIC (4C 8B C3) may not be strictly orthodox, but it seems to work all right.

INIT (hex 1930) and DATA (hex 193B)

This shorty reads in the constants needed, and stores them in page zero. SCR refers to the PET screen, TEMP is a temporary working area to hold the new generation as it is evolved, and RCS is essentially a copy of the PET screen data, which I found to be necessary to avoid "snow" on the screen during read/write operations directly on the screen locations. Up, down, etc. are the offsets to be added or subtracted from an address to get all the neighbor addresses. The observant reader will note the gap in the addresses between some of the routines.

TMPSCR (hex 1970)

This subroutine quickly transfers the contents of Temp and dumps it to the screen, using a dot (81 dec) symbol for a live cell (a 1 in TEMP) and a space (32 dec) for the absence of a live cell (a 0 in TEMP).

SCRTMP (hex 198A)

This is the inverse of TMPSCR, quickly transferring (and encoding) data from the screen into TEMP.

RSTORE (hex 19A6)

This subroutine fetches the initial addresses (high and low) for the SCR, TEMP, and RCS memory spaces.

NXTADR (hex 19BD)

Since we are dealing with 1000 bytes of data, we need a routine to increment to the next location, check for page crossing (adding 1 to the high address when it occurs), and checking for the end. The end is signaled by returning a 01 in the accumulator, otherwise a 00 is returned via the accumulator.

TMPRCS (hex 19E6)

The RCS address space is a copy of the screen, used as mentioned before to avoid constant "snow" on the screen if the screen were being continually accessed. This subroutine dumps data from TEMP, where the new generation has been computed, to RCS.

GENER (hex 1A00)

We finally arrive at a subroutine where LIFE is actually generated. After finding out the number of neighbors of the current RCS data byte from NBRS, GENER checks for births (CMPIM $03 at hex addr. 1A0E) if the cell was previously unoccupied. If a birth does not occur, there is an immediate branch to GENADR (the data byte remains 00). If the cell was occupied (CMPIM 81 dec at hex 1A08), OCC checks for survival (CMPIM $03 at hex 1A1A and CMPIM $02 at hex 1A1E), branching to GENADR when these two conditions are met, otherwise the cell dies (LDAIM $00 at hex 1A22). The results are stored in TEMP for the 1000 cells.

NBRS (hex 1A2F)

NBRS is the subroutine that really does most of the work and where most of the speed could be gained by more efficient programming. Its job, to find the total number of occupied neighbors of a given RCS data location, is complicated by page crossing and edge boundaries. In the present version, page crossing is taken care of, but edge boundaries (left, right, top, and bottom of the screen) are somewhat "strange". Above the top line and below the bottom line are considered as sort of forbidden regions where there should practically always be no "life" (data in those regions are not defined by the program, but I have found that there has never been a case where 81's have been present (all other data is considered as "unoccupied" characters). The right and left edges are different, however, and lead to a special type of "geometry". A cell at either edge is not considered as special by NBRS, and so to the right of a right-edge location is the next sequential address. On the screen this is really the left edge location, and one line lower. The inverse is true, of course for left addresses of left-edge locations. Topologically, this is equivalent to a "helix". No special effects of this are seen during a simple LIFE evolution since it just gives the impression of disappearing off one edge while appearing on the other edge. For an object like the "spaceship" (see Scientific American articles), then, the path eventually would cover the whole LIFE arena. The fun comes in when a configuration spreads out so much that it spills over both edges, and interacts with itself. This, of course cannot happen in an infinite universe, so that some of the more complex patterns will not have the same fate in the present version of LIFE. Most of the "blinkers", including the "glider gun" come out OK.

This 40x25 version of LIFE can undoubtedly be made more efficient, and other edge algorithms could be found, but I chose to leave it in its original form as a benchmark for my first successfully executed program in writing machine language on the PET. One confession, however - I used the KIM-1 to debug most of the subroutines. Almost all of them did not run on the first shot! Without a good understanding of PET memory allocation particularly in page zero, I was bound to crash many times over, with no recovery other than pulling the plug. The actual BASIC program consisted of a POKING loop with many DATA statements (always save on tape before running!).

Although the LIFE program was designed for use on the PET (8K version), no references are made to PET ROM locations or subroutines, and except for MAIN and SUBROUTINE address, are fully relocatable. The PET screen addresses (8000 - 83E8 hex) are treated as RAM. For anyone (with a 6502-based system) trying to convert the PET program, the following points need to be watched:

1. The BLANK symbol = 20 hex
2. The DOT symbol = 51 hex
3. The OFFSETs in DATA must be set for the user's display.

[Editor's Note: This seems like an ideal program to convert to an APPLE II and MICRO would be happy to print a list of the required modifications and enhancements that someone develops.]

A Brief Introduction to the Game of Life

by Mike Rowe

One of the interesting properties of the game of LIFE is that such simple rules can lead to such complex activity. The simplicity comes from the fact that the rules apply to each individual cell. The complexity comes from the interactions between the individual cells. Each individual cell is affected by its eight adjacent neighbors, and nothing else.

The rules are:

1. A cell survives if it has two or three neighbors.

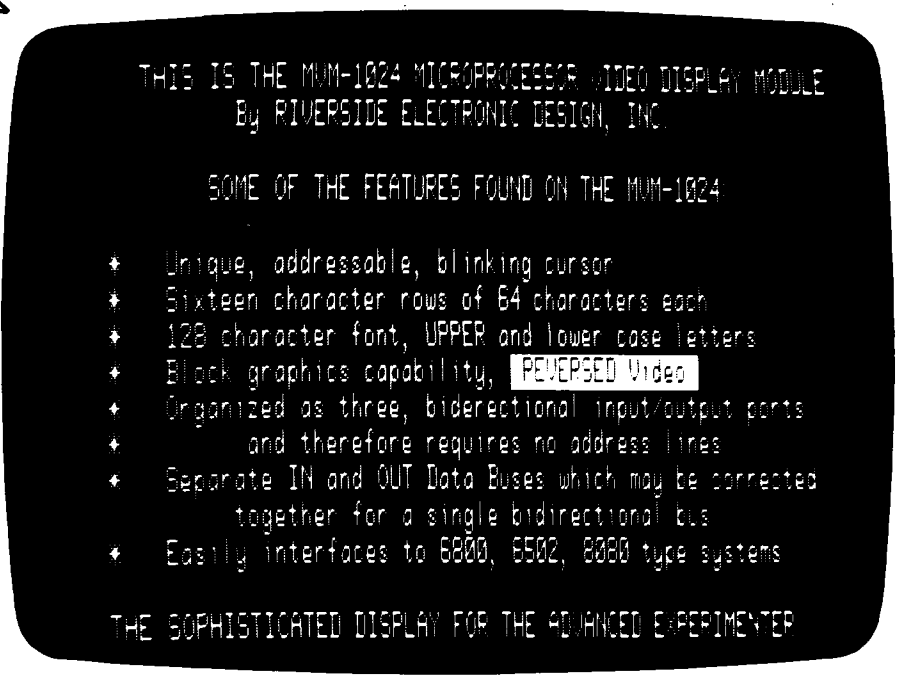
2. A cell dies from overcrowding if it has four or more neighbors. It dies from isolation if it has one or zero neighbors.

3. A cell is born when an empty space has exactly three neighbors.

With these few rules, many different types of activity can occur. Some patterns are STABLE, that is they do not change at all. Some are REPEATERS, patterns which undergo one or more changes and return to the original pattern. A REPEATER may repeat as fast as every other generation, or may have a longer period. A GLIDER is a pattern which moves as it repeats.

```
                                    REPEATERS
                              * *      *
       STABLE                                                  GLIDERS
                              * *        * *      * * *
                          * *        * *      * * *                         *
         *
                          * *            *                   *                  *
* *    *   *    *   * *
                                                               *    *           *
* *    *   *    * *   *
                                                           * * *      * * * * * *
         *
```

THIS IS WHAT YOUR KIM-1 CAN DO WITH RIVERSIDE'S KEM (KIM-1 EXPANSION MODULE) AND MVM-1024 VIDEO DISPLAY DRIVER. DISPLAY/MONITOR SOFTWARE AVALABLE IN EPROM (PLUGS INTO KEM) OR BURN YOUR OWN WITH OUR 2708/16 PROGRAMMER

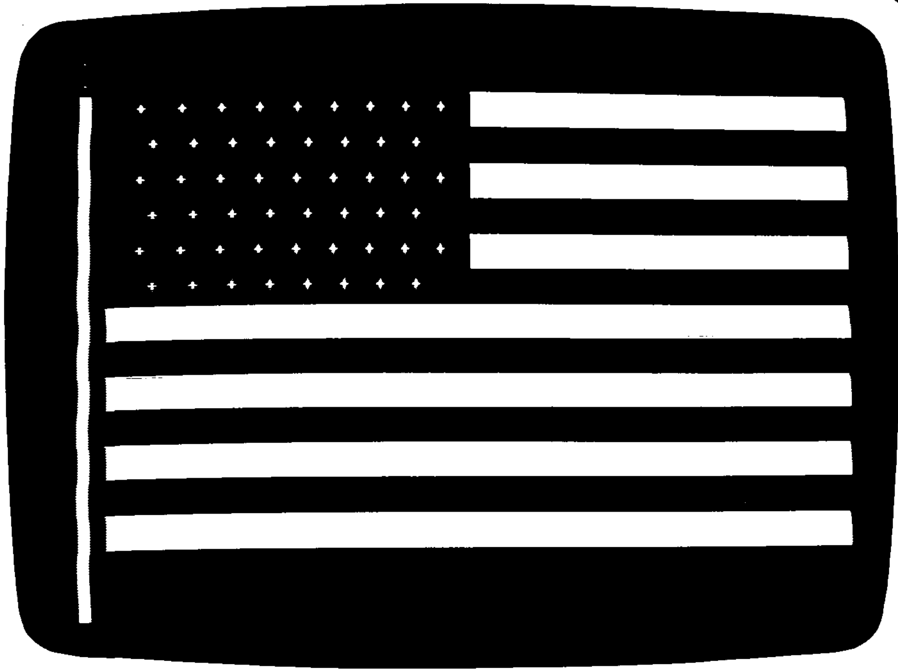


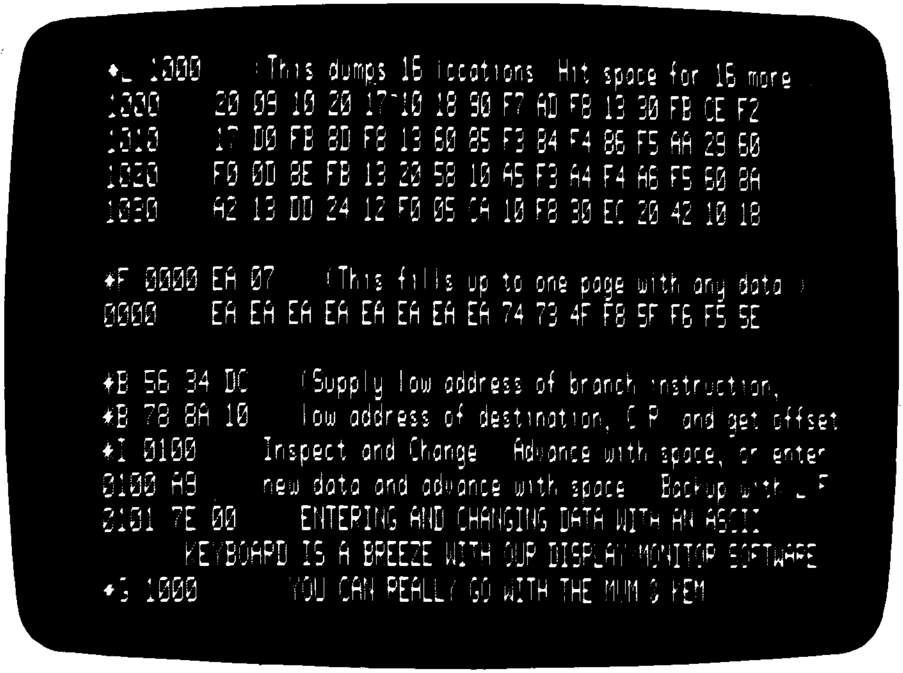
```
   THIS IS THE MVM-1024 MICROPROCESSOR VIDEO DISPLAY MODULE
          By RIVERSIDE ELECTRONIC DESIGN, INC.

        SOME OF THE FEATURES FOUND ON THE MVM-1024

 *   Unique, addressable, blinking cursor
 *   Sixteen character rows of 64 characters each
 *   128 character font, UPPER and lower case letters
 *   Block graphics capability, REVERSED Video
 *   Organized as three, biderectional input/output ports
 *          and therefore requires no address lines
 *   Separate IN and OUT Data Buses which may be corrected
          together for a single bidirectional bus
 *   Easily interfaces to 6800, 6502, 8080 type systems

THE SOPHISTICATED DISPLAY FOR THE ADVANCED EXPERIMENTER
```





```
*L 1000    (This dumps 16 locations  Hit space for 16 more
1000    20 09 10 20 17 10 18 90 F7 AD FB 13 30 FB CE F2
1010    17 D0 FB 8D F8 13 60 85 F3 84 F4 86 F5 AA 29 60
1020    F0 0D 8E FB 13 20 58 10 A5 F3 A4 F4 A6 F5 60 8A
1030    A2 13 DD 24 12 F0 05 CA 10 F8 30 EC 20 42 10 18

*F 0000 EA 07     (This fills up to one page with any data )
0000    EA EA EA EA EA EA EA EA 74 73 4F F8 5F F6 F5 5E

*B 56 34 DC    (Supply low address of branch instruction,
*B 78 8A 10     low address of destination, C P  and get offset
*I 0100       Inspect and Change  Advance with space, or enter
0100 A9       new data and advance with space  Backup with L F
0101 7E 00      ENTERING AND CHANGING DATA WITH AN ASCII
        KEYBOARD IS A BREEZE WITH OUR DISPLAY MONITOR SOFTWARE
*G 1000         YOU CAN REALLY GO WITH THE MVM & KEM
```

Riverside ELECTRONIC DESIGN INC.
1700 NIAGARA ST. BUFFALO, NY 14207

MVM-1024 $235; KEM $155; PROGRAMMER $75; EPROM $35
DIRECT FROM RIVERSIDE OR THROUGH YOUR DEALER

```
1900             LIFE   ORG   $1900

1900             BASIC  *     $C38B  RETURN TO BASIC ADDRESS
1900             OFFSET *     $002A  PAGE ZERO DATA AREA POINTER
1900             DOT    *     $0051  DOT SYMBOL = 81 DECIMAL
1900             BLANK  *     $0020  BLANK SYMBOL = 32 DECIMAL

1900             SCRL   *     $0020  PAGE ZERO LOCATIONS
1900             SCRH   *     $0021
1900             CHL    *     $0022
1900             CHH    *     $0023
1900             SCRLO  *     $0024
1900             SCRHO  *     $0025
1900             TEMPL  *     $0026
1900             TEMPH  *     $0027
1900             TEMPLO *     $0028
1900             TEMPHO *     $0029
1900             UP     *     $002A
1900             DOWN   *     $002B
1900             RIGHT  *     $002C
1900             LEFT   *     $002D
1900             UR     *     $002E
1900             UL     *     $002F
1900             LR     *     $0030
1900             LL     *     $0031
1900             N      *     $0032
1900             SCRLL  *     $0033
1900             SCRLH  *     $0034
1900             RCSLO  *     $0035
1900             RCSHO  *     $0036
1900             TMP    *     $0037
1900             TIMES  *     $0038
1900             RCSL   *     $0039
1900             RCSH   *     $003A

1900 08          MAIN   PHP          SAVE EVERYTHING
1901 48                 PHA          ON STACK
1902 8A                 TXA
1903 48                 PHA
1904 98                 TYA
1905 48                 PHA
1906 BA                 TSX
1907 8A                 TXA
1908 48                 PHA
1909 D8                 CLD          CLEAR DECIMAL MODE
190A 20 30 19           JSR   INIT
190D 20 8A 19           JSR   SCRTMP
1910 20 E6 19    GEN    JSR   TMPRCS
1913 20 00 1A           JSR   GENER
1916 20 70 19           JSR   TMPSCR
1919 E6 38              INCZ  TIMES  REPEAT 255 TIMES
191B D0 F3              BNE   GEN    BEFORE QUITTING
191D 68                 PLA          RESTORE EVERYTHING
191E AA                 TAX
191F 9A                 TXS
1920 68                 PLA
```

```
1921 A8               TAY
1922 68               PLA
1923 AA               TAX
1924 68               PLA
1925 28               PLP
1926 4C 8B C3         JMP   BASIC  RETURN TO BASIC

1930                  ORG   $1930

               MOVE VALUES INTO PAGE ZERO

1930 A2 19     INIT   LDXIM $19    MOVE 25. VALUES
1932 BD 3A 19  LOAD   LDAX  DATA   -01
1935 95 1F            STAZX $1F    STORE IN PAGE ZERO
1937 CA               DEX
1938 D0 F8            BNE   LOAD
193A 60               RTS

193B 00        DATA   =     $00    SCRL
193C 80               =     $80    SCRH
193D 00               =     $00    CHL
193E 15               =     $15    CHH
193F 00               =     $00    SCRLO
1940 80               =     $80    SCRHO
1941 00               =     $00    TEMPL
1942 1B               =     $1B    TEMPH
1943 00               =     $00    TEMPLO
1944 1B               =     $1B    TEMPHO
1945 D7               =     $D7    UP
1946 28               =     $28    DOWN
1947 01               =     $01    RIGHT
1948 FE               =     $FE    LEFT
1949 D8               =     $D8    UR
194A D6               =     $D6    UL
194B 29               =     $29    LR
194C 27               =     $27    LL
194D 00               =     $00    N
194E E8               =     $E8    SCRLL
194F 83               =     $83    SCRLH
1950 00               =     $00    RCSLO
1951 15               =     $15    RCSHO
1952 00               =     $00    TMP
1953 00               =     $00    TIMES

1970                  ORG   $1970

1970 20 A6 19  TMPSCR JSR   RSTORE GET INIT ADDRESSES
1973 B1 26     TSLOAD LDAIY TEMPL  FETCH BYTE FROM TEMP
1975 D0 06            BNE   TSONE  BRANCH IF NOT ZERO
1977 A9 20            LDAIM BLANK  BLANK SYMBOL
1979 91 20            STAIY SCRL   DUMP IT TO SCREEN
197B D0 04            BNE   TSNEXT
197D A9 51     TSONE  LDAIM DOT    DOT SYMBOL
197F 91 20            STAIY SCRL   DUMP IT TO SCREEN
1981 20 BD 19  TSNEXT JSR   NXTADR FETCH NEXT ADDRESS
1984 F0 ED            BEQ   TSLOAD
```

```
1986 20 A6 19        JSR   RSTORE RESTORE INIT ADDRESSES
1989 60              RTS

198A 20 A6 19 SCRTMP JSR   RSTORE GET INIT ADDRESSES
198D B1 20    STLOAD LDAIY SCRL   READ DATA FROM SCREEN
198F C9 51           CMPIM DOT    TEST FOR DOT
1991 F0 06           BEQ   STONE  BRANCH IF DOT
1993 A9 00           LDAIM $00    OTHERWISE ITS A BLANK
1995 91 26           STAIY TEMPL  STORE IT
1997 F0 04           BEQ   STNEXT UNCOND. BRANCH
1999 A9 01    STONE  LDAIM $01    A DOT WAS FOUND
199B 91 26           STAIY TEMPL  STORE IT
199D 20 BD 19 STNEXT JSR   NXTADR FETCH NEXT ADDRESS
19A0 F0 EB           BEQ   STLOAD
19A2 20 A6 19        JSR   RSTORE RESTORE INIT ADDRESSES
19A5 60              RTS

19A6 A9 00    RSTORE LDAIM $00    ZERO A, X, Y
19A8 AA              TAX
19A9 A8              TAY
19AA 85 20           STAZ  SCRL   INIT VALUES
19AC 85 26           STAZ  TEMPL
19AE 85 39           STAZ  RCSL
19B0 A5 25           LDAZ  SCRHO
19B2 85 21           STAZ  SCRH
19B4 A5 29           LDAZ  TEMPHO
19B6 85 27           STAZ  TEMPH
19B8 A5 36           LDAZ  RCSHO
19BA 85 3A           STAZ  RCSH
19BC 60              RTS

19BD E6 26    NXTADR INCZ  TEMPL  GET NEXT LOW ORDER
19BF E6 20           INCZ  SCRL   BYTE ADDRESS
19C1 E6 39           INCZ  RCSL
19C3 E8              INX
19C4 E4 33           CPXZ  SCRLL  IS IT THE LAST?
19C6 F0 0C           BEQ   PAGECH IS IT THE LAST PAGE?
19C8 E0 00           CPXIM $00    IS IT A PAGE BOUNDARY?
19CA D0 0E           BNE   NALOAD IF NOT, THEN NOT DONE
19CC E6 27           INCZ  TEMPH  OTHERWISE ADVANCE TO
19CE E6 21           INCZ  SCRH   NEXT PAGE
19D0 E6 3A           INCZ  RCSH
19D2 D0 06           BNE   NALOAD UNCONDITIONAL BRANCH
19D4 A5 34    PAGECH LDAZ  SCRLH  CHECK FOR LAST PAGE
19D6 C5 21           CMPZ  SCRH
19D8 F0 03           BEQ   NADONE IF YES, THEN DONE
19DA A9 00    NALOAD LDAIM $00    RETURN WITH A=0
19DC 60              RTS
19DD A9 01    NADONE LDAIM $01    RETURN WITH A=1
19DF 60              RTS

19E6                 ORG   $19E6

19E6 20 A6 19 TMPRCS JSR   RSTORE INIT ADDRESSES
19E9 B1 26    TRLOAD LDAIY TEMPL  FETCH DATA FROM TEMP
19EB D0 06           BNE   TRONE  IF NOT ZERO THEN ITS ALIVE
```

```
19ED A9 20           LDAIM BLANK  BLANK SYMBOL
19EF 91 39           STAIY RCSL   STORE IT IN SCREEN COPY
19F1 D0 04           BNE   NEWADR THEN ON TO A NEW ADDRESS
19F3 A9 51    TRONE  LDAIM DOT    THE DOT SYMBOL
19F5 91 39           STAIY RCSL   STORE IT IN SCREEN COPY
19F7 20 BD 19 NEWADR JSR   NXTADR FETCH NEXT ADDRESS
19FA F0 ED           BEQ   TRLOAD IF A=0, THEN NOT DONE
19FC 20 A6 19        JSR   RSTORE ELSE DONE. RESTORE
19FF 60              RTS

1A00 20 A6 19 GENER  JSR   RSTORE INIT ADDRESSES
1A03 20 2F 1A AGAIN  JSR   NBRS   FETCH NUMBER OF NEIGHBORS
1A06 B1 39           LDAIY RCSL   FETCH CURRENT DATA
1A08 C9 51           CMPIM DOT    IS IT A DOT?
1A0A F0 0C           BEQ   OCC    IF YES, THEN BRANCH
1A0C A5 32           LDAZ  N      OTHERWISE ITS BLANK
1A0E C9 03           CMPIM $03    SO WE CHECK FOR
1A10 D0 14           BNE   GENADR A BIRTH
1A12 A9 01    BIRTH  LDAIM $01    IT GIVES BIRTH
1A14 91 26           STAIY TEMPL  STORE IT IN TEMP
1A16 D0 0E           BNE   GENADR INCONDITIONAL BRANCH
1A18 A5 32    OCC    LDAZ  N      FETCH NUMBER OF NEIGHBORS
1A1A C9 03           CMPIM $03    IF IT HAS 3 OR 2
1A1C F0 08           BEQ   GENADR NEIGHBORS IT SURVIVES
1A1E C9 02           CMPIM $02
1A20 F0 04           BEQ   GENADR
1A22 A9 00    DEATH  LDAIM $00    IT DIED!
1A24 91 26           STAIY TEMPL  STORE IT IN TEMP
1A26 20 BD 19 GENADR JSR   NXTADR FETCH NEXT ADDRESS
1A29 F0 D8           BEQ   AGAIN  IF 0, THEN NOT DONE
1A2B 20 A6 19        JSR   RSTORE RESTORE INIT ADDRESSES
1A2E 60              RTS

1A2F 98       NBRS   TYA          SAVE Y AND X ON STACK
1A30 48              PHA
1A31 8A              TXA
1A32 48              PHA
1A33 A0 00           LDYIM $00    SET Y AND N = 0
1A35 84 32           STYZ  N
1A37 A2 08           LDXIM $08    CHECK 8 NEIGHBORS
1A39 B5 29    OFFS   LDAZX OFFSET -01
1A3B 10 15           BPL   ADD    ADD IF OFFSET IS POSITIVE
1A3D 49 FF           EORIM $FF    OTHERWISE GET SET TO
1A3F 85 37           STAZ  TMP    SUBTRACT
1A41 38              SEC          SET CARRY BIT FOR SUBTRACT
1A42 A5 39           LDAZ  RCSL
1A44 E5 37           SBCZ  TMP    SUBTRACT TO GET THE
1A46 85 22           STAZ  CHL    CORRECT NEIGHBOR ADDRESS
1A48 A5 3A           LDAZ  RCSH
1A4A 85 23           STAZ  CHH
1A4C B0 11           BCS   EXAM   OK, FIND OUT WHAT'S THERE
1A4E C6 23           DECZ  CHH    PAGE CROSS
1A50 D0 0D           BNE   EXAM   UNCOND. BRANCH
1A52 18       ADD    CLC          GET SET TO ADD
1A53 65 39           ADCZ  RCSL   ADD
1A55 85 22           STAZ  CHL    STORE THE LOW PART
```

```
1A57 A5 3A          LDAZ  RCSH   FETCH THE HIGH PART
1A59 85 23          STAZ  CHH
1A5B 90 02          BCC   EXAM   OK, WHAT'S THERE
1A5D E6 23          INCZ  CHH    PAGE CROSSING
1A5F B1 22    EXAM  LDAIY CHL    FETCH THE NEIGHBOR
1A61 C9 51          CMPIM DOT    DATA BYTE AND SEE IF ITS
1A63 D0 02          BNE   NEXT   OCCUPIED
1A65 E6 32          INCZ  N      ACCUMULATE NUMBER OF NEIGHBORS
1A67 CA       NEXT  DEX
1A68 D0 CF          BNE   OFFS   NOT DONE
1A6A 68             PLA          RESTORE X, Y FROM STACK
1A6B AA             TAX
1A6C 68             PLA
1A6D A8             TAY
1A6E 60             RTS
```

```
SYMBOL TABLE 2000 2186
BLANK  0020   SCRL   0020   SCRH   0021   CHL    0022
CHH    0023   SCRLO  0024   SCRHO  0025   TEMPL  0026
TEMPH  0027   TEMPLO 0028   TEMPHO 0029   OFFSET 002A
UP     002A   DOWN   002B   RIGHT  002C   LEFT   002D
UR     002E   UL     002F   LR     0030   LL     0031
N      0032   SCRLL  0033   SCRLH  0034   RCSLO  0035
RCSHO  0036   TMP    0037   TIMES  0038   RCSL   0039
RCSH   003A   DOT    0051   LIFE   1900   MAIN   1900
GEN    1910   INIT   1930   LOAD   1932   DATA   193B
TMPSCR 1970   TSLOAD 1973   TSONE  197D   TSNEXT 1981
SCRTMP 198A   STLOAD 198D   STONE  1999   STNEXT 199D
RSTORE 19A6   NXTADR 19BD   PAGECH 19D4   NALOAD 19DA
NADONE 19DD   TMPRCS 19E6   TRLOAD 19E9   TRONE  19F3
NEWADR 19F7   GENER  1A00   AGAIN  1A03   BIRTH  1A12
OCC    1A18   DEATH  1A22   GENADR 1A26   NBRS   1A2F
OFFS   1A39   ADD    1A52   EXAM   1A5F   NEXT   1A67
BASIC  C38B
```

```
SYMBOL TABLE 2000 2186
ADD    1A52   AGAIN  1A03   BASIC  C38B   BIRTH  1A12
BLANK  0020   CHH    0023   CHL    0022   DATA   193B
DEATH  1A22   DOT    0051   DOWN   002B   EXAM   1A5F
GENADR 1A26   GENER  1A00   GEN    1910   INIT   1930
LEFT   002D   LIFE   1900   LL     0031   LOAD   1932
LR     0030   MAIN   1900   N      0032   NADONE 19DD
NALOAD 19DA   NBRS   1A2F   NEWADR 19F7   NEXT   1A67
NXTADR 19BD   OCC    1A18   OFFS   1A39   OFFSET 002A
PAGECH 19D4   RCSH   003A   RCSHO  0036   RCSL   0039
RCSLO  0035   RIGHT  002C   RSTORE 19A6   SCRH   0021
SCRHO  0025   SCRL   0020   SCRLH  0034   SCRLL  0033
SCRLO  0024   SCRTMP 198A   STLOAD 198D   STNEXT 199D
STONE  1999   TEMPH  0027   TEMPHO 0029   TEMPL  0026
TEMPLO 0028   TIMES  0038   TMPRCS 19E6   TMPSCR 1970
TMP    0037   TRLOAD 19E9   TRONE  19F3   TSLOAD 1973
TSNEXT 1981   TSONE  197D   UL     002F   UP     002A
UR     002E
```

ROCKWELL'S NEW R6500/1

Rockwell International
Electronic Devices Division
3310 Miraloma Avenue
P.O. Box 3669
Anaheim, CA 92803

ANAHEIM, CA., May 11, 1978 -- A single-chip NMOS microcomputer (R6500/1) operating at 2 MHz with a 1 microsecond minimum instruction execution time, has been developed by Rockwell Int'l.

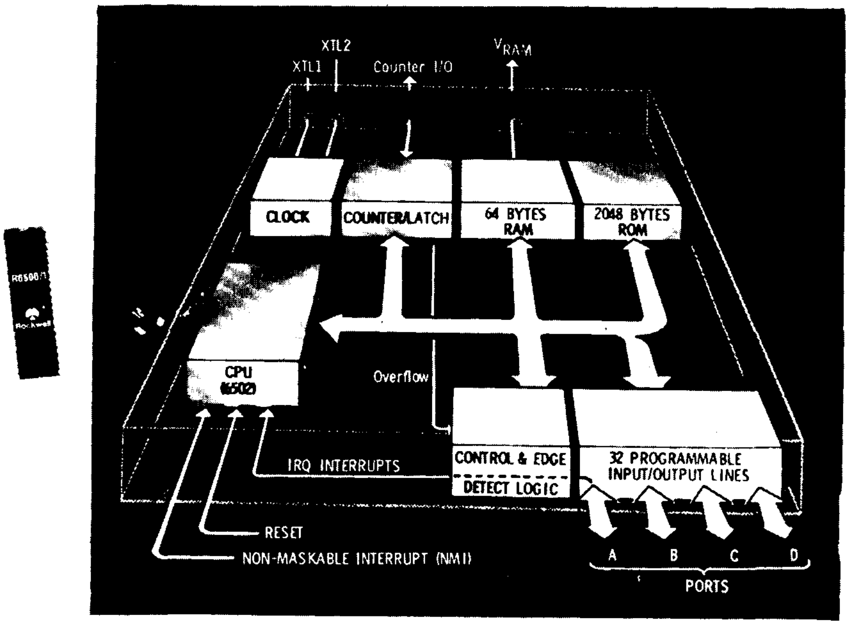
The 40-pin R6500/1 is fully software compatible with the 6500 family. It has the identical instruction set, including the 13 addressing modes, of the 6502 CPU. It operates from a single 5V power supply, and features a separate power pin which allows RAM memory to function on 10% of the operating power. On-chip features include 2K x 8 ROM, 64 x 8 RAM, 16-bit interval timer/event counter, and 32 bidirectional I/O lines. Additionally, it has maskable and non-maskable interrupts and an event-in/timer-out line.

The 32 bidirectional I/O lines are divided into four eight-bit ports (A, B, C and D). Each line can be selectively used as an input or an output. Two inputs to Port A can be used as edge sensing, software maskable, interrupt inputs -- one senses a rising edge; the other a falling edge.

Four different counter modes of operation are programmable: (1) free running with clock cycles counted for real time reference; (2) free running with output signal toggled by each counter overflow; (3) external event counter; and (4) pulse width measurement mode. A 16-bit latch automatically reinitializes the counter to a preset value. Interrupt on overflow is software maskable.

A 64-pin Emulator part, of which 40 pins are electrically identical to the standard R6500/1 part and which comes in either 1 MHz or 2 MHz versions, is available now. Rockwell expects to begin receiving codes from customers in July for production deliveries in Sept. Quantity prices for 6500/1 production devices are under $10.00 for both the 1 MHz and 2 MHz models. Single-unit prices for Emulator parts are $75.00 for the 1 MHz model and $95.00 for the 2 MHz version.

Contact: Leo Scanlon - 714/632-2321
Pattie Atteberry - 213/386-8600



ONE-CHIP SPEEDSTER -- Functional diagram of one-chip NMOS microcomputer (R6500/1) developed by Rockwell International. Fully software compatible with the 6500 family, the R6500/1 operates from a single 5V power supply at 2 MHz with a 1 microsecond minimum execution time.

6502 INTERFACING FOR BEGINNERS: ADDRESS DECODING I

Marvin L. De Jong
Dept. of Math-Physics
The School of the Ozarks
Point Lookout, MO 65726

This is the first installment of a column which will appear on a regular basis as long as reader interest, author enthusiasm and the editor's approval exist. Your response will be vital for our deciding whether to continue the column. Do not be afraid to be critical or to make suggestions about what subjects you would like to see. Hopefully, the column will be of interest to anyone who owns a 6502 system. One of the more challenging aspects of being a computer hobbyist is understanding how your system works and being able to configure and construct I/O ports. Then one can begin to tie his computer to the outside world. Perhaps this column will give you the ability to produce flashing lights, clicking relays, whirring motors, and other remarkable phenomena to amaze your friends and make your mother proud.

An educational column has to make some assumptions about where the readers are in terms of their understanding. A familiarity with binary and hex numbers will be assumed, as will a nodding acquaintance with the 7400 series of integrated circuits. Lacking such a background I would recommend that you get a book like "Bugbook V" by Rony, Larsen, and Titus; "TTL Cookbook" by Lancaster; or an equivalent book from your local computer shop or mail order house. Ads in "Micro", "Byte", "Kilobaud", "Ham Radio", "73 Magazine", etc. will list places where both books and parts may be ordered. My own preference for "hands-on" experience would be "Bugbook V". Although this book has some material on the 8080A chip, most of the material is very general and the chapters covering the basic 7400 series integrated circuits are very good. Another indispensable book is the "TTL Data Book" published by Texas Instruments.

It would be a good idea to get a Proto Board or equivalent breadboarding system for the experiments which will be suggested. One can even find wire kits to go with the breadboards. I would not purchase all the Outboards from E & L Instruments since the same circuits can be constructed less expensively from parts. Please regard these suggestions as opinions which may not be shared by all experimenters.

Finally, let me introduce the column by saying that the title is not "Interfacing Made Easy". If it were easy there would be no challenge and no need for this column. Like mountain climbing, satisfaction comes from overcoming the difficult rather than achieving the obvious. The material which you see in this column will usually be something which I am in the process of learning myself. I am a hobbyist like yourselves: I keep the wolf from the door by teaching mathematics and physics, not computer science or digital electronics. Expert opinions from readers and guest contributions will always be welcome.

We begin at the beginning. The 6502 pins may be divided into four groups: power, address, data, and control pins. Pins 1 and 21 are grounds, and pin 8 is connected to the +5V supply, making the power connections. Pins 9 through 20 and 22 through 25 are connected to the address bus on the microcomputer, while the data pins, 26 through 33, are connected to the data bus. All of the remainder of the pins may be lumped in the general class of control pins. In subsequent issues the data bus and the control bus will be discussed. Our concern in the first two issues is with addressing.

The 6502 Address Bus

The 6502 receives data from a variety of devices (memory, keyboard, tape reader, floppy disc, etc.), processes it, and sends it back to one or more devices. The first process is called READ and is accomplished by the LDA or similar instruction. The last process is called WRITE and is achieved by a STA type instruction. The purpose of the address pins is to put out a signal on the address bus to select the device or location which is going to produce or accept the data. In the computer system, each device has a unique address, and when the 6502 puts that address on the address bus, the device must be activated. Each line on the address bus may have one of two possible values (high or low, H or L, 1 or 0, +5V or 0V are the names most frequently given to these values). A one-address-line system could select two devices; one activated by a 0 on the address line, the other by a 1. Figure 1 shows how to decode such an idiot microcomputer.

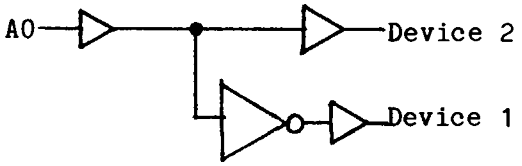


Figure 1. Decoding a One-Address Line Microprocessor.

Any device which when connected to the address bus puts out a unique signal (1 or 0) for a unique address is called a decoder. We have seen that a microcomputer with a single address line can select two devices, which could be memory locations or I/O ports. A somewhat smarter microprocessor might have two address lines. It could be decoded by the device shown in Figure 2, provided the truth table of the device were the one given in Table 1. Such a device could be implemented with NAND OR NOR gates, or with a 74139.

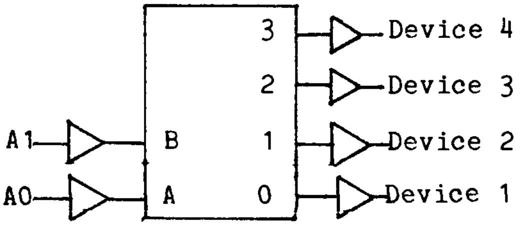


Figure 2. 74139 Decoder for a Two-Address Line Microprocessor.

Inputs		Outputs			
A	B	0	1	2	3
L	L	L	H	H	H
L	H	H	L	H	H
H	L	H	H	L	H
H	H	H	H	H	L

Table 1. Truth Table for Two-Line Decoder 74139.

The point is that two address lines allow the microprocessor to select four devices; three address lines give eight devices; four, 16; five, 32; six, 64; and so on. The 6502, being very smart, has 16 address lines. Anyone who can calculate how many telephones can be "addressed" by a 7-digit, base-ten phone number can also calculate how many locations can be addressed by a 16 digit, base-two address bus, The answers are 10^7=10 million and 2^{16}=65,536, respectively.

Earth people have not yet made a single device to simultaneously decode 16 address lines to produce 65,536 device select signals. Such a monster IC would need at least 65,554 pins. Many integrated circuits are constructed to decode the ten, low-order address lines (A0-A9) internally. For example, the 6530 PIA chips on the KIM and the 21L02 memory chips on my memory board decode the ten lowest address lines internally, that is, they select any one of the 2^{10} =1024 flip-flops to be written to or read. Consequently, our problem is to decode the high-order address lines, at least initially. These lines are usually decoded to form blocks of address space (not unlike home addresses in city blocks). Three address lines give eight (2^3=8) possible blocks, and the three highest address lines (A15-A13) divide the address space into eight blocks, each having $2^{(16-3)}$ =2^{13} locations.

Now 1024 (1024=2^{10}) locations is usually referred to as 1K, so 2^{13} locations is 2^3 x 2^{10} locations, which is 8 x 2^{10} locations, which is 8K locations. Thus the top three address lines divide the address space into eight, 8K blocks. See Table 2 for more details. Each of these 8K blocks may be further divided

A15	A14	A13	Name	Hex Addresses
0	0	0	8K0	0000-1FFF
0	0	1	8K1	2000-3FFF
0	1	0	8K2	4000-5FFF
0	1	1	8K3	6000-7FFF
1	0	0	8K4	8000-9FFF
1	0	1	8K5	A000-BFFF
1	1	0	8K6	C000-DFFF
1	1	1	8K7	E000-FFFF

Table 2. "Blocking" the Memory Space.

into 1K blocks by decoding address lines A12-A10. Table 3 shows how block 8K4 is divided into eight, 1K blocks. Finally, as mentioned before, many devices decode the lowest 10 address lines, and consequently we have decoded all 16 address lines, at least on paper.

A12	A11	A10	Name	Hex Address
0	0	0	K32	8000-83FF
0	0	1	K33	8400-87FF
0	1	0	K34	8800-8BFF
0	1	1	K35	8C00-8FFF
1	0	0	K36	9000-93FF
1	0	1	K37	9400-97FF
1	1	0	K38	9800-9BFF
1	1	1	K39	9C00-9FFF

Table 3. Subdivision of 8K4 Block into 1K blocks.

To begin to see how this is done, construct the circuit shown in Figure 3.

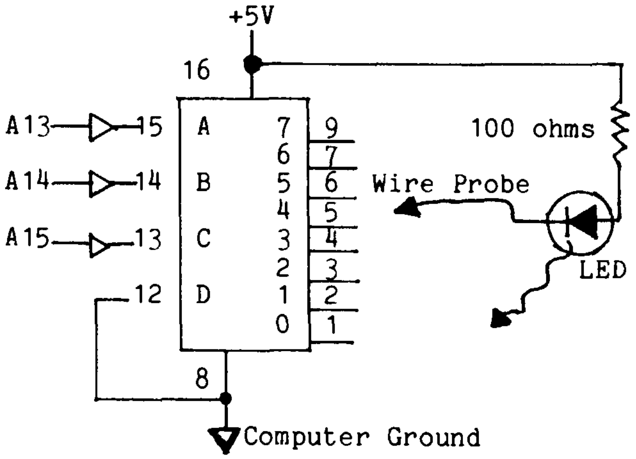


Figure 3. Decoding the Highest Three Address Lines.

(There are many decoding schemes and circuits, the circuit of Figure 3 is just one possible technique.) Here is where your breadboard becomes useful. Connect the address lines from your 6502 system to the 74145. (KIM owners can do this with no buffering because lines A15-A13 are not used on the KIM-1. Owners of other systems should check to see if the address lines are properly buffered.) Now perform the following experiments:

1. Load the following program somewhere between 0100 and 1FFF. The program is relocatable.

```
0200  18                CLC
0201  8D XX 60   LOOP   STA 60XX
0204  90 FB             BCC LOOP
```

This routine stores Accum. in location 60XX. X means "don't care." Then loop back.

2. Run the program and with the wire probe shown in Figure 3, test each of the output pins (pins 1-7 and 9). Which ones cause the LED to glow?

3. Try to explain your results with the help of the truth table, Table 4.

4. Change the STA instruction to a LDA instruction (AD XX 60) and repeat steps 2 and 3 above.

5. In turn, change the location at which you are getting the data to a location in each of the 8K blocks in Table 2, e.g. 00XX, 20XX, 40XX, etc. and test the output pins on the 74145 to see if the LED glows. You should be able to explain your results with the truth table.

6. Stop the program and check the pins again.

Inputs			Outputs							
C	B	A	0	1	2	3	4	5	6	7
L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H
L	H	L	H	H	L	H	H	H	H	H
L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	H	H	H	L	H	H	H
H	L	H	H	H	H	H	H	L	H	H
H	H	L	H	H	H	H	H	H	L	H
H	H	H	H	H	H	H	H	H	H	L

Table 4. Truth Table for 74LS145 when connected as shown in Figure 3.

In steps 2 and 4 the LED should glow when the probe touches pin 1 and pin 4. Why does it glow more brightly on pin 1? When the program is stopped, only pin 1 should cause the LED to light. The answers to these questions and the answers to questions you never asked will be given in the next issue.

What else is coming up in the next column? We will see how to take any of the 8 signals from the 74145 to enable a 74LS138 which in turn will decode address lines A12-A10, thus dividing any 8K block of address space which we may select into 1K blocks. Into one of these 1K blocks we will put some I/O ports.

(The more precocious of my attentive readers may already see that the scheme of Figure 3 could also be used to preset or clear a flip-flop to control an external device, for example, a heater, and all that without even using the data lines. If you see all that, you can take over this column.) See you next issue.

HALF A WORM IN THE APPLE

Mike Rowe
P.O. Box 3
S. Chelmsford, MA 01824

Last issue we reported a potential problem that had been discovered in the Apple II, relating to using PIA'a. The problem had been uncovered by the staff of EDN in the course of developing a system based on an Apple II board. The matter is not totally resolved, but the following is what we have heard.

I called Steve Wozniak of Apple and asked about the problem. He said that he had sent a chip to EDN which had cleared up the problem. He did not indicate that there was any more to it.

I then talked to John Conway of EDN. He maintained that a problem still does exist with Apple II interfacing to 6520 or 6522 PIAs. It can be done, but requires the addition of a chip to slow down the phase 0 signal to make it the equivalent of the phase 2 signal. The PIA can not be directly interfaced, as would normally be expected in a 6502-based system. John stated that the chip required costs about $7.00.

Another angle on the picture was also reported to me by John. He had found a company on the West Coast that is making interfaces for the Apple II. The engineer there had discovered the same problem.

There is a fairly complete discussion of the problem and the solution in the May 20, 1978 edition of EDN. If anyone has additional information to shed on the situation, MICRO will be happy to publish it. The problem does not seem to be all that serious, and we do not want to dwell on it, but we hope that this discussion has prevented some of our readers from going nuts trying to add a PIA to their Apple II.

EDN BLASTS THE 6502

Robert M. Tripp
P.O. Box 3
S. Chelmsford, MA 01824

The May 20, 1978 issue of EDN which had the information on the Apple II/PIA, ended with a "put down" of the 6502, by Jack Hemenway. I feel that the attack, and that is what I would call it, was a very emotional one, based on the fact that the author has worked with the 6800 extensively. His points were such "fatal flaws" in the 6502 as:

> the stack is limited to page 1
> the index registers are 8-bit
> the two different methods of
>   indirect indexing are confusing
> there are too many addressing modes
> there is only one accumulator
>   and so forth.

Of course we can all think of things that we would like to have in a micro, but there have to be trade-offs, and a lot of people seem to be happy with the 6502's set of capabilities. I suggest that some of us write to EDN and advise them of the 6502's good points. For example, I prefer the stack to be only in page one. I have written a lot of code and have never used up very much of the stack. And, if a program goes wild, only page one is destroyed - not all of memory. So, let us set EDN straight by writing a few letters. The editor has said he would be happy to hear from us.

ROCKWELL'S AIM IS PRETTY GOOD

Rockwell International
Microelectronic Devices
P.O. Box 3669
Anaheim, CA 92803
714/632-3729

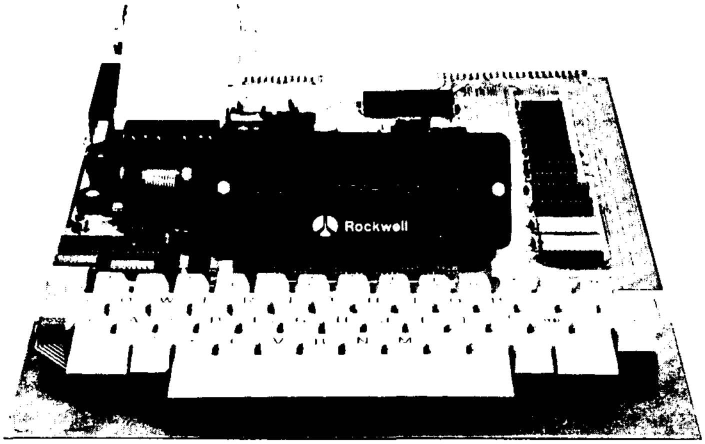
Rockwell's AIM 65 (Advanced Interface Module) gives you an assembled, versatile microcomputer system with a full-size keyboard, 20-character display and a 20-character thermal printer!

AIM 65's terminal-style ASCII keyboard has 54 keys providing 69 different alphabetic, numeric and special functions.

AIM 65's 20-character true Alphanumeric Display uses 16-segment font monolithic characters that are both unambiguous and easily readable.

AIM 65's 20-column Thermal Printer prints on low-cost heat sensitive roll paper at a fast 90 lines per minute. It produces all the standard 64 ASCII characters with a crisp-printing five-by-seven dot matrix. AIM 65's on-board printer is a unique feature for a low cost computer.

The CPU is the R6502 operating a 1 MHz. The basic system comes with 1K RAM, expandable on-baord to 4K. It includes a 4K ROM Monitor, and can be expanded on-board to 16K using 2332 ROMs or can also accept 2716 EPROMs. An R6532 RAM-Input/Output-Timer is used to support AIM 65 functions. There are also two R6522 Versatile Interface Adaptors. Each VIA has two 8-bit, bidirectional TTL ports, two 2-bit peripheral handshake control ports and two fully programmable interval timer/counters.

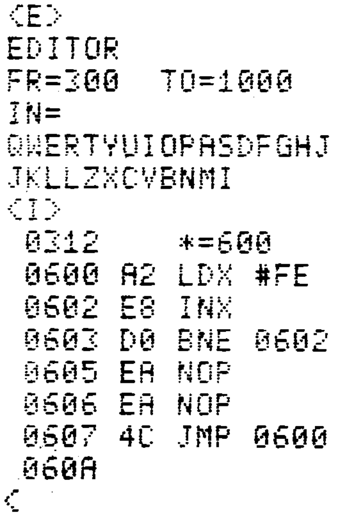


The built-in expansion capability includes a 44-pin Application Connector for peripheral add-ons and a 44-pin Expansion Connector with the full system bus. And, both connectors are totally KIM-1 compatible!

TTY and Audio Cassette Interfaces are part of the basic system. There is a 20 ma current loop TTY interface, just like the KIM-1, and an Audio Cassette Interface which has a KIM-1 compatible format as well as its own special binary blocked file assembler compatible format.

The DEBUG/MONITOR includes a mini-assembler and a text editor. Editing may use the keyboard, TTY, cassette, printer and display. The Monitor includes a typical set of memory display/modify commands. It also has peripheral device controllers, breakpoint capability and single step/trace modes of debugging. An 8K BASIC Interpreter will be available in ROM as an option.

AIM 65 will be available in August. It will cost $375.



```
<E>
EDITOR
FR=300  TO=1000
IN=
QWERTYUIOPASDFGHJ
JKLLZXCVBNMI
<I>
 0312     *=600
 0600 A2 LDX #FE
 0602 E8 INX
 0603 D0 BNE 0602
 0605 EA NOP
 0606 EA NOP
 0607 4C JMP 0600
 060A
<
```

IS IT TIME TO RENEW YOUR SUBSCRIPTION?

If you are a subscriber to MICRO, then the two digit code following your name on the mailing is the number of the last issue your current subscription covers. If your two digit code is 05, then this is your last issue. And, you original subscribers with an 06 will be up for renewal soon. MICRO will NOT be sending out reminders. So, if your number is coming up, get your subscription renewal in soon.

MICRO is published bi-monthly. The first issue was OCT/NOV 1977. Single copy price is $1.50. Subscriptions are $6.00 per year, 6 issues, in the USA. One year subscriptions to other countries are: [Payment must be in US $.]

Surface:	Canada/Mexico	$7.00
	All Other Countries	$8.00
Air Mail:	Europe	$14.00
	South America	$14.00
	Central America	$12.00
	All Other Countries	$16.00

Name: . . . .

Addr: . . . .

City: . . . .

State: . . . . . . Zip: . . . .

Amount: . . . .

Country: . . . .

Issues #1, 2, and 3 have been reprinted, so that back issues are now available for all issues. The price is $1.50 per copy - USA, Canada or Mexico. Other countries add $.50 per copy surface or $1.25 per copy air mail.

Your name and address will be made available to legitimate dealers, suppliers, and other 6502 interests so that you may be kept informed of new products, current developments, and so forth, unless you specify that you do not wish your name released to these outside sources.

Send payment to: MICRO, P.O. Box 3, S. Chelmsford, MA 01824, USA

INFORMATION FOR ADVERTISERS

If you are interested in reaching the 6502 market, consider advertising in MICRO. Since MICRO is devoted to the 6502, its readers are actively interested in 6502 related products and will pay attention to your material. Your ad will not be lost among ads for 8080's, Z-80's, etc. Since the content of MICRO is primarily useful and factual reference material, each issue will be referred to many times, giving your ad multiple exposure. The cost of advertising in MICRO is quite low. Current rates are: Full Page $100.00, Half page $60.00, Quarter page $35.00. It is easy to place your ad. Provide camera ready copy either one-to-one or two-to-one in size. Photographs should be glossy and one-to-one or two-to-one in size. Payment must accompany the ad unless credit terms have been previously established. Our current circulation is over 2000: 1400+ subscriptions and 700+ to dealers. Dealers report that MICRO sells very well. One dealer who specializes in 6502 products reports that MICRO outsells Kilobaud! Bulk rates to dealers are $.90 per copy with a minimum of ten copies.

DEADLINES: August/September Issue: Ad reservation - July 10. Ad copy - July 17.
October/November Issue: Ad reservation - Sept 11. Ad copy - Sept 18.

To reserve your ad, or for further information, call Judy at 617/256-3649.
Mailing address: MICRO Ads, P.O. Box 3, S Chelmsford, MA 01824.

A SLOW LIST FOR APPLE BASIC

Bob Sander-Cederlof
8413 Midpark Road #3072
Dallas, TX 75240

One of the nicest things about Apple BASIC is its speed. It runs circles around most other hobby systems! Yet there are times when I honestly wish it were a little slower.

Have you ever typed in a huge program, and then wanted to review it for errors? You type "LIST", and the whole thing flashes past your eyes in a few seconds! That's no good, so you list it piecemeal -- painfully typing in a long series like:

```
        LIST 0,99
        LIST 100,250
           .
           .
           .
        LIST 21250,21399
```

As the reviewing and editing process continues, you have to type these over and over and over . . . Ouch!

At the March meeting of the Dallas area "Apple Corps" several members expressed the desire to be able to list long programs slowly enough to read, without the extra effort of typing separate commands for each screen-full. One member suggested appending the series of LIST commands to the program itself, with a subroutine to wait for a carriage return before proceeding from one screen-full to the next. For example:

```
9000 LIST 0,99:GOSUB 9500
9010 LIST 100,250: GOSUB 9500
          .
          .
          .
9250 LIST 21250,21399:GOSUB 9500
9260 END
9500 INPUT A$:RETURN
```

While this method will indeed work, it is time-consuming to figure out what line ranges to use in each LIST command. It is also necessary to keep them up-to-date after adding new lines or deleting old ones.

But there is a better way! I wrote a small machine language program which solves our problem. After this little 64-byte routine is loaded and activated the LIST command has all the features we wanted.

1. The listing proceeds at a more leisurely pace, allowing you to see what is going by.
2. The listing can be stopped temporarily, by merely pressing the space bar. When you are ready, pressing the space bar a second time will cause the listing to resume.
3. The listing can be aborted before it is finished, by typing a carriage return.

The routine as it is now coded resides in page three of memory, from $0340 to $037F. It is loaded from cassette tape in the usual way: *340.37FR.

After the routine is loaded, you return to BASIC. The slow-list features are activated by typing "CALL 887". They may be de-activated by typing "CALL 878" or by hitting the RESET key.

How does it work? The commented assembly listing should be self-explanatory, with the exception of the tie-in to the Apple firmware. All character output in the Apple funnels through the same subroutine: COUT, at location $FDED. The instruction at $FDED is JMP ($0036) This means that the address which is stored in locations $0036 and $0037 indicates where the character output subroutine really is. Every time you hit the RESET key, the firmware monitor sets up those two locations to point to $FDF0, which is where the rest of the COUT subroutine is located. If characters are supposed to go to some other peripheral device, you would patch in the address of your device handler at these same two locations. In the case of the slow-list program, the activation routine merely patches locations $0036 and $0037 to point to $0340. The de-activation routine makes them point to $FDF0 again.

Every time slow-list detects a carriage return being output, it calls a delay subroutine in the firmware at $FCA8. This has the effect of slowing down the listing. Slow-list also keeps looking at the keyboard strobe, to see if you have typed a space or a carriage return. If you have typed a carriage return, slow-list stops the listing and jumps back into BASIC at the soft entry point ($E003). If you have typed a space, slow-list goes into a loop waiting for you to type another character before resuming the listing.

That is all there is to it! Now go turn on your Apple, type in the slow-list program, and list to your heart's content!

```
0340                    ORG   $0340

                  ROUTINE TO SLOW DOWN APPLE BASIC LISTINGS

0340 C9 8D     SLOW    CMPIM $8D     CHECK IF CHAR IS CARRIAGE RETURN
0342 D0 1A             BNE   CHROUT  NO, SO GO BACK TO COUT
0344 48                PHA           SAVE CHARACTER ON STACK
0345 2C 00 C0          BIT   $C000   TEST KEYBOARD STROBE
0348 10 0E             BPL   WAIT    NOTHING TYPED YET
034A AD 00 C0          LDA   $C000   GET CHARACTER FROM KEYBOARD
034D 2C 10 C0          BIT   $C010   CLEAR KEYBOARD STROBE
0350 C9 A0             CMPIM $A0     CHECK IF CHAR IS A SPACE
0352 F0 10             BEQ   STOP    YES - STOP LISTING
0354 C9 8D             CMPIM $8D     CHECK IF CHAR IS A CARRIAGE RETURN
0356 F0 09             BEQ   ABORT   YES - ABORT LISTING
0358 A9 00     WAIT    LDAIM $00     MAKE A LONG DELAY
035A 20 A8 FC          JSR   $FCA8   CALL MONITOR DELAY SUBROUTINE
035D 68                PLA           GET CHARACTER FROM STACK
035E 4C F0 FD  CHROUT  JMP   $FDF0   REJOIN COUT SUBROUTINE
0361 4C 03 E0  ABORT   JMP   $E003   SOFT ENTRY INTO APPLE BASIC
0364 2C 00 C0  STOP    BIT   $C000   WAIT UNTIL KEYBOARD STROBE
0367 10 FB             BPL   STOP    APPEARS ON THE SCENE
0369 8D 10 C0          STA   $C010   CLEAR THE STROBE
036C 30 EA             BMI   WAIT    UNCONDITIONAL BRANCH

                  SUBROUTINE TO DE-ACTIVATE SLOW LIST

036E A9 F0     OFF     LDAIM $F0     RESTORE $FDF0 TO
0370 85 36             STAZ  $36     LOCATIONS 36 AND 37
0372 A9 FD             LDAIM $FD
0374 85 37             STAZ  $37
0376 60                RTS

                  SUBROUTINE TO ACTIVATE SLOW LIST

0377 A9 40     ON      LDAIM $40     SET $0340 INTO
0379 85 36             STAZ  $36     LOCATIONS 36 AND 37
037B A9 03             LDAIM $03
037D 85 37             STAZ  $37
037F 60                RTS

SYMBOL TABLE
ABORT  0361   CHROUT 035E   OFF    036E   ON     0377
SLOW   0340   STOP   0364   WAIT   0358


SYMBOL TABLE
SLOW   0340   WAIT   0358   CHROUT 035E   ABORT  0361
STOP   0364   OFF    036E   ON     0377
```

THE MICRO SOFTWARE CATALOG: II

Mike Rowe
P.O. Box 3
S. Chelmsford, MA 10824

Name: ZZYP-PAX for PET, #1,2, and 3
System: PET
Memory: 8K RAM
Language: BASIC
Hardware: Standard PET
Description: Each of these three ZZYP-for PET includes a cassette with two games and a booklet designed to educate the beginning or intermediate level PET programmer. #1 has IRON PLANET (Rescue the Princess) and HANGMAN (Guess the secret word). Included is a 12 page booklet which not only contains game rules, but has 5 pages of useful programming techniques including: Direct Screen Access Graphics, Flashing Messages, and Programmed Delays. #2 contains BLACK BART (a mean-mouthed poker player) and BLACK BRET (for blackjack - one or two players). #3 contains BLOCK and FOOTBALL both of which allow either two-player or play-the-PET options.
Copies: Just released, 40 copies.
Price: $9.95 each
Includes: PET tape cassette, instructions and educational manual with info for program modifications.
Ordering Info: Specify ZZYP-PAX number
Author: Terry Dossey
Available from:
Many PET dealers, or,
ZZYP Data Processing
2313 Morningside Drive
Bryan, TX 77801

Name: BULLS AND BEARS (tm)
System: Apple II
Memory: 16K
Language: 16K BASIC
Hardware: Apple II
Description: A multi-player simulation of corporate finance. Involves decision-making regarding production levels, financing, dividends, buying and selling of stock, etc.
Copies: "Hundreds sold"
Price: $12.00
Includes: Game cassette and booklet.
Ordering Info: At computer stores only
Author: SPEAKEASY SOFTWARE LTD.
Box I200
Kemptville, Ontario
Canada KOG 1JO

[Dealer inquiries invited]

Name: A Variety of Programs
System: Apple II
Memory: Most 8K or less
Language: Mostly Integer BASIC
Hardware: Mostly standard Apple II
Description: A varied collection of short programs. Some utilities, some educational. Included are: ALPHA SORT MUSIC ROUTINE, STOP WATCHBASIC DUMP, MULTIPLY, ONE-ARM-BANDIT, ...
Copies: Varies, up to about 20.
Price: $7.50 to $10.00 each.
Includes: Apple II cassette and program listing.
Ordering Info: Write for catalog.
Author(s): Not specified.
Available from:
Apple PugetSound Prog. Lib. Exch.
6708 39th Avenue SW
Seattle, WA 98136

Name: HELP Information Retrieval
System: KIM-1
Memory: Basic KIM-1
Language: Assembler and HELP
Hardware: KIM-1, terminal, cassettes
Description: Permits the user to create a data base on cassette, and then perform a variety of searches on the data base. May make six simultaneous tests on FLAGS associated with the data plus one test on each of the six data fields. Permits very complex retrieval from the data base. Includes ULTRATAPE which reads/writes at 100 char/sec, 12 times the normal KIM rate.
Copies: 100+
Price: $15.00
Includes: Cassette tape, 36 page User Manual, a Source Listing book and a Functions Manual which explains the operation of the HELP language.
Ordering Info: Specify HELP Info Ret.
Author: Robert M. Tripp
Available from:
Many 6502 Dealers, or,
The COMPUTERIST, Inc.
P.O. Box 3
S. Chelmsford, MA 01824

BEEPER BLOOPER AND OTHER MICROBES

We apologize to the many readers who have experienced problems trying to get the simple "KIM Beeper" to work. There was an error in the listing. The cause of the error was trivial; the effect was devastating! "A KIM BEEPER" by Gerald C. Jenkins appeared in issue #4, on page 43. The corrected listing is given below, in full. You would have to examine the alphabetic portion of the two listings quite closely to see error. The line at address 0118 read:

```
   BIT   TIME
```

but should have read:

```
   BIT   TIMER
```

A minor error, only one letter missing, but look at the difference in the listings from that point on. A two byte instruction was generated instead of the correct three bytes. This, in addition to being wrong, caused every subsequent location to be displaced by one byte.

In this case, the error was our fault. We try to check the listings presented in MICRO, but we do not have the equipment or time to run every program. We have caught some errors in programs submitted to us, and we test what we can.

There was a slight bug in "A Complete Morse Code Send/Receive Program for the KIM-1" by Marvin L. De Jong. The second line of the listing read:

```
0057 A9 FF              LDAIM $FF
```

but should have been:

```
0057  A9 40             LDAIM $40
```

The only effect this will have will be to set an incorrect initial code speed.

In "An Apple II Programmer's Guide" by Rick Auricchio, the paragraph which states that "control K, followed by 5" sets the keyboard to device 5, is in error. It is really "5, followed by control K".

```
0100                  ORG   $0100

0100           TIME   *     $00FF
0100           NOTE   *     $00C8
0100           PBD    *     $1702
0100           PBDD   *     $1703
0100           TIMER  *     $1707

0100 A9 FF     BEEP   LDAIM TIME   START TIMER FOR 1/4 SECOND TONE
0102 8D 07 17         STA   TIMER  USING INTERVAL TIMER
0105 A9 01            LDAIM $01    SET OUTPUT TONE OFF
0107 8D 02 17         STA   PBD
010A 8D 03 17         STA   PBDD

010D 4D 02 17  TONE   EOR   PBD    TOGGLE OUTPUT
0110 8D 02 17         STA   PBD
0113 A0 C8            LDYIM NOTE   SET TO COUNT FOR NOTE LENGTH
0115 88        TONEX  DEY          $C8 = 500 HZ
0116 D0 FD            BNE   TONEX  CYCLE IN DOWN COUNTER
0118 2C 07 17         BIT   TIMER  TEST 1/4 SECOND UP
011B 10 F0            BPL   TONE   CONTINUE TONE IF NOT DONE
011D A9 01            LDAIM $01    TURN TONE OFF
011F 8D 02 17         STA   PBD
0122 A9 FF            LDAIM TIME   START WAIT BETWEEN BEEPS
0124 8D 07 17         STA   TIMER
0127 2C 07 17  NOTONE BIT   TIMER  WAIT FOR TIME OUT
012A 10 FB            BPL   NOTONE
012C CA               DEX          DECREMENT NUMBER OF BEEPS COUNTER
012D D0 D1            BNE   BEEP   ANOTHER BEEP OR DONE
012F 60               RTS          RETURN TO CALLING ROUTINE
```

A BASIC 6502 DISASSEMBLER
FOR APPLE AND PET

Michael J. McCann
28 Ravenswood Terrace
Cheektowaga, NY 14225

A disassembler is a program that accepts machine language (object code) as input and produces a symbolic representation that resembles an assembler listing. Although disassemblers have a major disadvantage viz., that they cannot reproduce the labels used by the original programmer, they can prove very useful when one is attempting to transplant machine code programs from one 6502 system to another. This article describes a disassembler program written in Commodore BASIC.

The disassembler (see listing and sample run) uses the mnemonics listed in the Oct-Nov 1977 issue of MICRO. The output is in this format: (address) (byte#1) (byte#2) (byte#3) (mnemonic) (bytes #2 and #3)

The address is outputted in decimal (base 10). The contents of the byte(s) making up each instruction are printed in hexadecimal (base 16) between the address and the mnemonic. In three byte instructions the high order byte is multiplied by 256 and added to the contents of the low order byte, giving the decimal equivalent of the absolute address. This number is printed in the (bytes #2 and #3) field. In two byte instructions the decimal equivalent of the second byte is printed in the (bytes #2 and #3) field.

Programming Comments

Lines 10-40 initialize the BY% and MN$ arrays (BY% contains the number of bytes in each instruction and MN$ contains the mnemonic of each instruction)

Lines 60-80 initialize the decimal to hexadecimal conversion array (CO$)

Lines 100-130 input the starting address

Lines 1000-1050 decimal to hexadecimal conversion subroutine

Lines 3000-5030 do the disassembly

Lines 3010-3030 take care of illegal operation codes

Line 3050 transfers control to one of three disassembly routines, the choice is determined by the number of bytes in the instruction

Lines 6000-6290 contain the data for the arrays

Although this was originally written in Commodore BASIC, it will work with the APPLESOFT BASIC of the APPLE computer.

```
SAMPLE RUN

RUN

START ADDRESS

? 64004

64004  4C 7E E6 JMP  59006

64007  AD 0A 02 LDA  522

64010  F0 08    BEQ  8

64012  30 04    BMI  4
```

```
  1 REM A 6502 DISASSEMBLER
  2 REM BY MICHAEL J. MCCANN
  3 REM WILL RUN ON AN 8K PET OR AN APPLE WITH APPLESOFT BASIC
  10 DIM MN$(256)BY%(256),CO$(16)
  20 FOR E=0 TO 255
  30 READ MN$(E),BY%(E)
  40 NEXT E
  60 FOR E=0 TO 15
  70 READ CO$(E)
  80 NEXT E
  100 PRINT CHR$(147)
  110 PRINT:PRINT "START ADDRESS"
  120 INPUT AD
* 130 PRINT
  140 I=0
  150 GOTO 3000
  1000 SX=INT(DC/16)
  1010 UN=DC-(SX*16)
  1020 SX$=CO$(SX)
  1030 UN$=CO$(UN)
  1040 HX$=SX$+UN$
  1050 RETURN
  3000 IF I=16 THEN 5050
  3005 I=I+1
  3010 IB=PEEK(AD)
  3015 IF MN$(IB)<>"NULL" GOTO 3050
  3020 IB=DC:GOSUB 1000
  3030 PRINT AD;TAB(8);HX$;"*"
  3035 AD=AD+1
  3040 GOTO 5030
  3050 ON BY%(IB) GOTO 3060,3090,4050
  3060 DC=IB:GOSUB 1000
  3070 PRINT AD;TAB(8);HX$;TAB(17);MN$(IB)
  3075 AD=AD+1
  3080 GOTO 5030
  3090 DC=IB:GOSUB 1000
  4000 B1$=HX$
  4010 DC=PEEK(AD+1):GOSUB 1000
  4020 B2$=HX$
  4030 PRINT AD;TAB(8);B1$;" ";B2$;TAB(17);MN$(IB);TAB(21);PEEK(AD+1)
  4035 AD=AD+2
  4040 GOTO 5030
  4050 DC=IB:GOSUB 1000
  4060 B1$=HX$
  4070 DC=PEEK(AD+1):GOSUB 1000
  4080 B2$=HX$
  4090 DC=PEEK(AD+2):GOSUB 1000
  5000 B3$=HX$
  5010 OP=PEEK(AD+1)+(PEEK(AD+2)*256)
  5020 PRINT AD;TAB(8);B1$;" ";B2$;" ";B3$;TAB(17);MN$(IB);TAB(21);OP
  5025 AD=AD+3
  5030 GOTO 3000
  5050 INPUT A
* 5060 PRINT
  5070 I=0
  5080 GOTO 3000
```

Note: The two PRINT statements with an * are required by APPLESOFT to prevent the first output line from being mis-aligned. They may not be required by the PET BASIC.

```
6000 DATA BRK,1,ORAIX,2,NULL,0,NULL,0,NULL,0,ORAZ,2,ASL,2,NULL,0,PHP,1
6010 DATA ORAIM,2,ASLA,1,NULL,0,NULL,0,ORA,3,ASL,3,NULL,0,BPL,2,ORAIY,2
6020 DATA NULL,0,NULL,0,NULL,0,ORAZX,2, ASLZX,2,NULL,0,CLC,1,ORAY,3
6030 DATA NULL,0,NULL,0,NULL,0,ORAX,3,ASLX,3,NULL,0,JSR,3,ANDIX,2,NULL,0
6040 DATA NULL,0,BITZ,2,ANDZ,2,ROLZ,2,NULL,0,PLP,1,ANDIM,2,ROLA,1,NULL,0
6050 DATA BIT,3,AND,3,ROL,3,NULL,0,BMI,2,ANDIY,2,NULL,0,NULL,0,NULL,0
6060 DATA ANDZX,2,ROLZX,2,NULL,0,SEC,1,ANDY,3,NULL,0,NULL,0,NULL,0,ANDX,3
6070 DATA ROLX,3,NULL,0,RTI,1,EORIX,2,NULL,0,NULL,0,NULL,0,EORZ,2,LSRZ,2
6080 DATA NULL,0,PHA,1,EORIM,2,LSRA,1,NULL,0,JMP,3,EOR,3,LSR,3,NULL,0
6090 DATA BVC,2,EORIY,2,NULL,0,NULL,0,NULL,0,EORZX,2,LSRZX,2,NULL,0
6100 DATA CLC,1,EORY,3,NULL,0,NULL,0,NULL,0,EORX,3,LSRX,3,NULL,0,RTS,1
6110 DATA ADCIX,2,NULL,0,NULL,0,NULL,0,ADCZ,2,RORZ,2,NULL,0,PLA,1,ADCIM,2
6120 DATA RORA,1,NULL,0,JMI,3,ADC,3,ROR,3,NULL,0,BVS,2,ADCIY,2,NULL,0
6130 DATA NULL,0,NULL,0,ADCZX,2,RORZX,2,NULL,0,SEI,1,ADCY,3,NULL,0,NULL,0
6140 DATA NULL,0,ADCX,3,RORX,3,NULL,0,NULL,0,STAIX,2,NULL,0,NULL,0,STYZ,2
6150 DATA STAZ,2,STXZ,2,NULL,0,DEY,1,NULL,0,TXA,1,NULL,0,STY,3,STA,3
6160 DATA STX,3,NULL,0,BCC,2,STAIY,2,NULL,0,NULL,0,STYZX,2,STAZX,2,STXZY,2
6170 DATA NULL,0,TYA,1,STAY,3,TXS,1,NULL,0,NULL,0,STAX,3,NULL,0,NULL,0
6180 DATA LDYIM,2,LDAIX,2,LDXIM,2,NULL,0,LDYZ,2,LDAZ,2,LDXZ,2,NULL,0
6190 DATA TAY,1,LDAIM,2,TAX,1,NULL,0,LDY,3,LDA,3,LDX,3,NULL,0,BCS,2
6200 DATA LDAIY,2,NULL,0,NULL,0,LDYZX,2,LDAZX,2,LDXZY,2,NULL,0,CLV,1
6210 DATA LDAY,3,TSX,1,NULL,0,LDYX,3,LDAX,3,LDXY,3,NULL,0,CPYIM,2,CMPIX,2
6220 DATA NULL,0,NULL,0,CPYZ,2,CMPZ,2,DECZ,2,NULL,0,INY,1,CMPIM,2,DEX,1
6230 DATA NULL,0,CPY,3,CMP,3,DEC,3,NULL,0,BNE,2,CMPIY,2,NULL,0,NULL,0
6240 DATA NULL,0,CMPZX,2,DECZX,2,NULL,0,CLD,1,CMPY,3,NULL,0,NULL,0,NULL,0
6250 DATA CMPX,3,DECX,3,NULL,0,CPXIM,2,SBCIX,2,NULL,0,NULL,0,CPX,2,SBCZ,2
6260 DATA INCZ,2,NULL,0,INX,1,SBCIM,2,NOP,1;NULL,0,CPX,3,SBC,3,INC,3
6270 DATA NULL,0,BEQ,2,SBCIY,2,NULL,0,NULL,0,NULL,0,SBCZX,2,INCZX,2,NULL,0,SED,1
6280 DATA SBCY,3,NULL,0,NULL,0,NULL,0,SBCX,3,INCX,3,NULL,0
6290 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
```

SYNERTEK'S VIM-1

Synertek Incorporated
P.O. Box 552
Santa Clara, CA 95052

Synertek has announced a new 6502-based microcomputer system with the following features:

FULLY-ASSEMBLED AND COMPLETELY INTEGRATED SYSTEM that's ready-to use as soon as you open the box.

28 DOUBLE-FUNCTION KEYPAD INCLUDING UP TO 24 "SPECIAL" FUNCTIONS.

EASY-TO-VIEW 6-DIGIT HEX LED DISPLAY.

KIM-1 HARDWARE COMPATIBILITY.

The powerful 6502 8-bit MICROPROCESSOR whose advanced architectural features have made it one of the largest selling "micros" on the market today.

THREE ON-BOARD PROGRAMMABLE INTERVAL TIMERS available to the user for timing loops, watchdog functions, and real-ime communication protocols.

4K BYTE ROM RESIDENT MONITOR and Operating Programs.

Single 5 Volt power capability is all that is required.

1K BYTES OF 2114 STATIC RAM on-board with sockets provided for immediate expansion to 4K bytes on-board, with total memory expansion to 65,536 bytes.

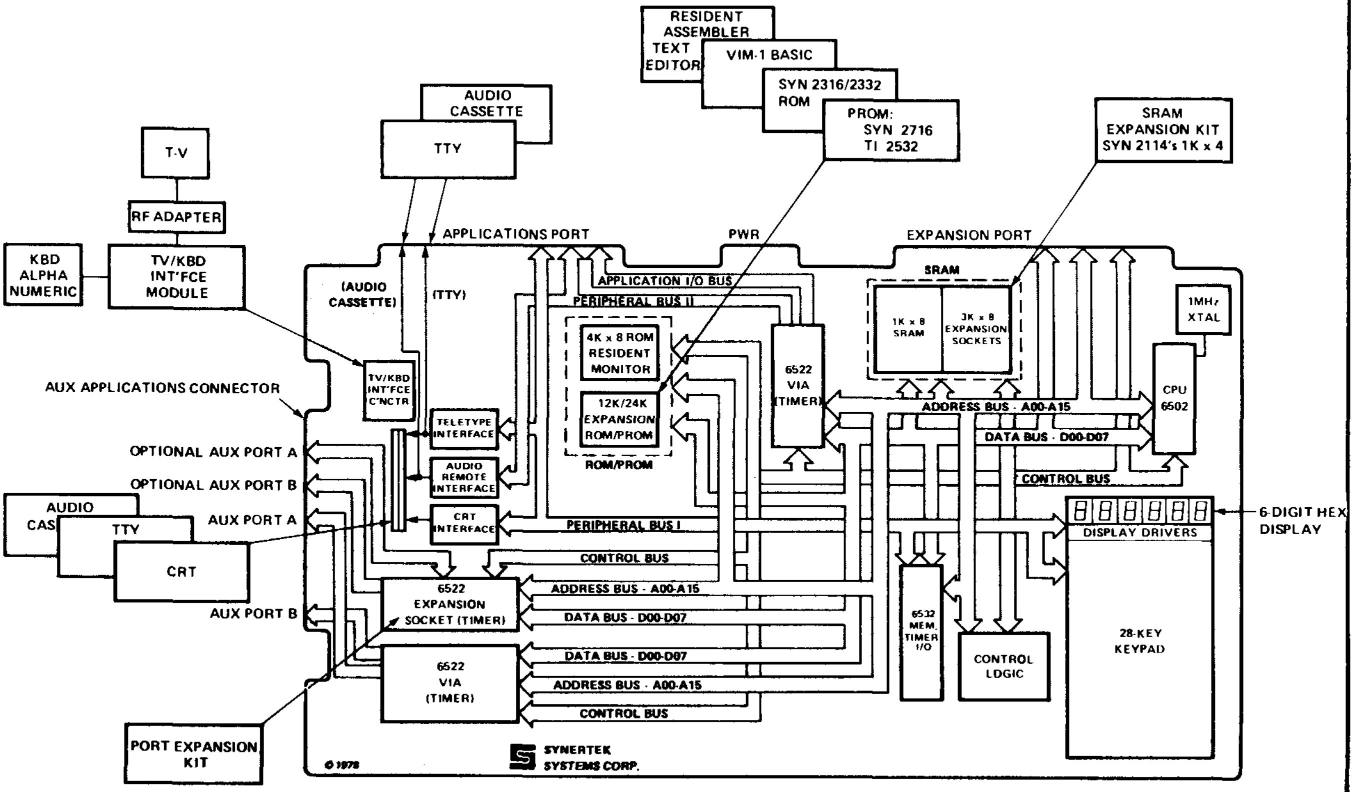
USER PROM/ROM: The system is equipped with 3 PROM/ROM expansion sockets for 2316/2332 ROMs or 2716 EPROMs.

ENHANCED SOFTWARE with simplified user interface.

STANDARD INTERFACES INCLUDE:
- Audio Cassette Recorder Interface with Remote Control (Two modes: 135 Baud KIM-1 compatible, Hi-speed 2400 Baud).
- Full Duplex 20mA Teletype Interface
- System Expansion Bus Interface
- TV Controller Board Interface
- CRT Compatible Interface

APPLICATION PORT: 15 Bi-directional TTL lines for user applications with expansion capability for added lines.

EXPANSION PORT FOR ADD-ON MODULES (50 I/O Lines in the basic system).



APPLAYER MUSIC INTERPRETER

Richard F. Suitor
166 Tremont Street
Newton, MA 02158

There have been several routines for making music with the APPLE II, including one in MICRO and one in the APPLE documentation. The program described here is more than a tone-making routine, it is a music interpreter. It enables one to generate a table of bytes that specify precisely the half-tone and duration of a note with a simple coding. Its virtue over the simpler routines is similar to that of any interpreter (such as Sweet 16, or, more tenuously, BASIC) over an assembler or hand coding - it is easier to achieve one's goal and easier to decipher the coding six months later.

The immediate motivation for this interpreter was Martin Gardner's Mathematical Games Column in the April 1978 Scientific American. Several types of algorithmically generated music are discussed in that column; this program provides a means of experimenting with them as well as a convenient method of generating familiar tunes.

The program is written in 6502 assembly language. It would be usable on a system other than the APPLE if a speaker were interfaced in a similar way. Accessing a particular address (C030) changes the current through the APPLE speaker from on to off or from off to on; it acts like a push button on/off switch (or, of course, a flip-flop). Thus this program makes sound by accessing this address periodically with an LDA C030. Any interface that could likewise be activated with a similar (4 clock cycles) instruction could be easily used. A different interfacing software procedure would change the timing and require more extensive modification.

The tone is generated with a timing loop that counts for a certain number of clock cycles, N (all of the cycles in a period including the toggling of the speaker are counted). Every N cycles a 24 bit pattern is rotated and the speaker is toggled if the high order bit is set. Four cycles are wasted (to keep time) if the bit is not set. There is a severe limit to the versatility of a waveshape made from on/off transitions, but tones resembling a variety of (cheap) woodwinds and pipes are possible, with fundamentals ranging from about 20 Hz to 8 KHz.

Applayer interprets bytes to produce different effects. There are two types of bytes:

Note bytes	Bit 7 Not Set
Control bytes	Bit 7 Set to 1

A note byte enables one to choose a note from one of 16 half tones, and from one to eight eighth notes in duration. The low order nybble is the half-tone; the high order nybble is the duration (in eighth notes) minus one.

Bit	7	6	5	4	3	2	1	0
Note Byte	0	(Duration)			(Half-Tone)			

The control bytes enable one to change the tempo, the tonal range which the 16 half-tones cover, rests, the waveshape of the tone and to jump from one portion of the table to another.

Control Byte Table

HEX	DECIMAL	FUNCTION
81	129	The next three bytes are the new waveshape pattern
82	130	JMP - New table address follows. Low order byte first , then page byte
83	131	JSR - new table address follows. When finished, continuing this table at byte after address byte
9N	144+N	N is the number of 16th notes to be silent at the tail of a note. Controls rests and note definition
AN	160+N<32	Selects the tonal range. Half-tone #0 is set to one of 32 half-tones giving a basic range of four octaves
CN	192+N<62	Controls the tempo. Length of a note is proportional to N. Largest value gives a whole note lasting about 3.5 sec.
FF	255	RETURN. Stop interpreting this table. Acts as return for 83 JSR instruction or causes return from Applayer.

To use Applayer with sheet music, one must first decide on the range of the half tones. This must sometimes be changed in the middle of the song. For example, the music for "Turkey in the Straw", which appears later, was in the key of C; for the first part of the song I used the following table.

NOTE	C	D	E	F	G	A	B	C	D
TONE #	0	2	4	5	7	9	B	C	E

The tonal range was set with a control byte, B0. In the chorus, the range of the melody shifts up; there the tonal range is set with a B7 and the table is

NOTE	G	A	B	C	D	E	F	G	A
TONE#	0	2	4	5	7	9	A	C	E

(The actual key is determined by the wave shape pattern as well as the tonal range control byte. For the pattern used, 05 05 05, the fundamental for the note written as C would be about 346Hz, which is closer to F.)

Rests can be accomplished with a 9N control byte and a note byte. For example, 94 10 is a quarter rest, 98 30 is a half rest etc. This control is normally set at 91 for notes distinctly separated, or to 90 for notes that should run together.

Let's try to construct a table that Applayer can use to play a tune. We can start simply with "Twinkle, Twinkle Little Star". That tune has four lines the first and fourth are identical, as are the second and third. So our table will be constructed to:

1. Set up the tonal range, tone pattern and tempo that we want
2. JSR to a table for the first line
3. JSR to a table for the second line
4. Repeat #3
5. Repeat #2
6. Return
7. First line table and return
8. Second line table and return

Since unfortunately Applayer is not symbolic, it will be easier to construct the tables in reverse, so that we can know where to go in steps 2-6. The note table for the first line can go at 0B00 and looks like:

```
0B00-  10 10 17 17 19 19 37 15
0B08-  15 14 14 12 12 30 FF FF
```

The second line can follow at 0B10:

```
0B10-  17 17 15 15 14 14 32 FF
```

Now we can start on step 1. I'll suggest the following to start; you'll want to make changes:

```
0B20-  B0 81 05 05 05 E0 91
```

The above determines the tonal range, the tone wave shape, the tempo, and a sixteenth note rest out of every note to keep the notes distinct. To run them together, use 90 instead of 91. Steps 2 - 6 can follow immediately:

```
0B20-                          83
0B28-  00 0B 83 10 0B 83 10 0B
0B30-  83 00 0B FF
```

That completes the table for "Twinkle, Twinkle". We now have to tell Applayer where it is and turn it on. From BASIC we must set up some zero page locations first and then JSR to Applayer:
(Don't forget to set LOMEM before running; 2900 will do for this table.)

```
100 POKE 19,32   (low order byte of the
                  table address, 0B20)
110 POKE 20,11   (high order byte of the
                  table address, 0B20)
120 POKE 1,8     (high order byte of 1st
                 pg of Applayer program)
130 POKE 17,8     (16 & 17 contain the
                 tone table address)
140 POKE 16,0
120 CALL 2346     (jump subroutine to
                  092A)
```

We can also make a short program in assembly language to set up the zero page locations. See routine ZERO, location 09C0 in the listing.

This initialization can be used most easily by reserving the A00 page, or much of it, as a "Table of Contents" for the various note tables elsewhere in memory. To do this with "Twinkle, Twinkle" we add the following table:

```
0A20-  02 20 0B
```

Which jumps immediately to the table at 0B20. With this convention, we can move from table to table by changing only the byte at 9D0 (2512 decimal).

We can use this initialization from BASIC, too, by changing the last instruction to RTS:

```
100 POKE 2512,32  LOW ORDER TABLE BYTE
110 POKE 2538,96  CHANGE INST. AT 09EA
120 CALL 2496     TO RTS.
```

From the monitor:

```
*9D0:20
*9C0G
```

will do.

If, as I, you quickly tire of "Twinkle, Twinkle", you may wish to play with "Turkey in the Straw". The table follows; its structure will be left as an exercise.

From the monitor:

```
*9D0:0
*9C0G
```

will play it.

```
0A00:  03 90 0F 83 90 0F FF

0F00:  90 1C 1A 92 38 90 18 1A
0F08:  18 13 10 11 91 13 13 33
0F10:  33 90 18 1A 92 3C 3C 90
0F18:  1C 1A 18 1A 91 1C 38 10
0F20:  38 90 1C 1A 92 38 90 18
0F28:  1A 18 13 91 10 11 13 53
0F30:  33 90 18 1A 91 3C 3F 90
0F38:  1F 1C 18 1A 1C 18 92 3A
0F40:  94 78 91 FF
0F50:  01 55 55 55 FF
0F58:  01 05 05 05 FF
0F60:  15 18 18 15 78 FF
0F68:  16 1A 1A 16 7A FF
0F70:  1D 1D 1D 1D 18 18 18 18
0F78:  35 15 15 33 90 11 13 91
0F80:  15 18 18 18 90 18 15 11
0F88:  13 91 15 15 13 13 71 FF
0F90:  03 58 0F D4 B0 83 00 0F
0F98:  B7 83 60 0F 83 50 0F 83
0FA0:  60 0F 83 50 0F 83 68 0F
0FA8:  83 50 0F 83 68 0F 83 50
0FB0:  0F 83 70 0F FF
```

Tone Table

```
0800:  A0 03 68 03 38 03 08 03
0808:  E0 02 B8 02 90 02 68 02
0810:  48 82 28 02 08 02 E8 01
0818:  D0 01 B4 01 9C 01 84 01
0820:  70 01 5C 01 48 01 34 01
0828:  24 01 14 01 04 01 F4 00
0830:  E8 00 DA 00 CE 00 C2 00
0838:  B8 00 AE 00 A4 00 9A 00
0840:  92 00 8A 00 82 00 7A 00
0848:  74 00 6D 00 67 00 61 00
0850:  5C 00 57 00 52 00 4D 00
0858:  49 00 45 00 41 00 3D 00
```

```
                 APPLAYER MUSIC INTERPRETER

                 R. F. SUITOR   APRIL 1978

                 TIMING LOOP
                 LOCATIONS 0 THROUGH 7 ARE SET BY CALLING ROUTINE
                 8 CYCLE LOOP TIMES Y REG PLUS 0-7 CYCLES
                 DETERMINED BY ENTRY POINT

0860                    ORG    $0860

0860 EA          TIME   NOP
0861 EA                 NOP
0862 EA                 NOP
0863 88          TIMEA  DEY
0864 85 45              STA    $0045  ANY INNOCUOUS 3 CYCLE INSTRUCTION
0866 D0 FB              BNE    TIMEA  BASIC 8 CYCLE LOOP
0868 F0 05              BEQ    TIMEC
086A 88          TIMEB  DEY
086B EA                 NOP
086C EA                 NOP
086D D0 F4              BNE    TIMEA
086F 24 04       TIMEC  BIT    $0004  START CHECK OF BIT PATTERN
0871 38                 SEC           IN 2, 3, AND 4
0872 30 02              BMI    TIMED
0874 EA                 NOP
0875 18                 CLC
0876 26 02       TIMED  ROL    $0002
0878 26 03              ROL    $0003
087A 26 04              ROL    $0004
087C 90 03              BCC    TIMEE
087E AD 30 C0           LDA    $C030  TOGGLE SPEAKER
0881 C6 06       TIMEE  DEC    $0006  DURATION OF NOTE IN
0883 D0 05              BNE    TIMEF  NO. OF CYCLES IN LOCATIONS
0885 C6 07              DEC    $0007  6 AND 7
0887 D0 05              BNE    TIMEG
0889 60                 RTS
088A EA          TIMEF  NOP           TIMING EQUALIZATION
088B EA                 NOP
088C D0 00              BNE    TIMEG
088E A4 05       TIMEG  LDY    $0005
0890 6C 00 00           JMI    $0000

                 SCALING ROUTINE FOR CYCLE DURATION
                 CALCULATION  LOC 6,7 = A REG * LOC 50,51

0893 85 45       SCALE  STA    $0045
0895 A9 00              LDAIM  $00
0897 85 06              STA    $0006
0899 85 07              STA    $0007
089B A2 05              LDXIM  $05
089D 18                 CLC
089E 66 07       SCALEX ROR    $0007
08A0 66 06              ROR    $0006
08A2 46 45              LSR    $0045
08A4 90 0C              BCC    SCALEA
```

```
08A6 A5 06               LDA    $0006
08A8 65 50               ADC    $0050
08AA 85 06               STA    $0006
08AC A5 07               LDA    $0007
08AE 65 51               ADC    $0051
08B0 85 07               STA    $0007
08B2 CA          SCALEA  DEX
08B3 10 E9               BPL    SCALEX
08B5 E6 07               INC    $0007  DUE TO SIMPLE LOGIC IN TIMING ROUTINE
08B7 60                  RTS

08BE                     ORG    $08BE

                 NOTE PLAYING ROUTINE
                 Y REG HAS HALF-TONE INDEX

08BE A5 12       NOTE    LDA    $0012  NOTE LENGTH
08C0 85 52               STA    $0052
08C2 A5 0F               LDA    $000F  NOTE TABLE OFFSET
08C4 85 10               STA    $0010
08C6 B1 10               LDAIY  $0010  LOW ORDER BYTE OF MACHINE
08C8 38                  SEC           CYCLES PER PERIOD
08C9 85 54               STA    $0054
08CB E9 35               SBCIM  $35    CYCLES USED UP TIMING OVERHEAD
08CD 85 08               STA    $0008
08CF C8                  INY
08D0 B1 10               LDAIY  $0010  HIGH ORDER BYTE OF MACHINE
08D2 85 55               STA    $0055  CYCLES PER PERIOD
08D4 E9 00               SBCIM  $00
08D6 85 09               STA    $0009
08D8 A9 00               LDAIM  $00
08DA 85 50               STA    $0050
08DC 85 51               STA    $0051
08DE 85 53               STA    $0053
08E0 A0 10               LDYIM  $10
08E2 20 86 FB            JSR    $FB86

                 THIS PART IS PARTICULAR TO APPLE.  THE DIVIDE
                 ROUTINE AT FB86 IS USED.  OR, PROVIDE A ROUTINE
                 WHICH DIVIDES LOCS 54,55 BY 52,53 AND LEAVES THE
                 RESULT IN 50,51 FOR THE SCALING ROUTINE.

08E5 A5 08               LDA    $0008
08E7 48                  PHA
08E8 46 09               LSR    $0009
08EA 6A                  RORA
08EB 46 09               LSR    $0009
08ED 6A                  RORA
08EE 46 09               LSR    $0009
08F0 6A                  RORA
08F1 85 05               STA    $0005  NO. OF 8 CYCLE LOOPS
08F3 68                  PLA
08F4 29 07               ANDIM  $07    LEFT OVER CYCLES DETERMINT
08F6 AA                  TAX           ENTRY POINT
08F7 BD F8 09            LDAX   TTABLE TABLE OF ENTRY POINTS FOR TIMING LOOP
08FA 85 00               STA    $0000
```

```
08FC A5 0E              LDA   $000E  NOTE DURATION, QUARTER, HALF
08FE 38                 SEC
08FF E5 0D              SBC   $000D  REST PART OF NOTE
0901 F0 0F              BEQ   NOTEB  IF NOTHING TO DO
0903 20 93 08           JSR   SCALE  SCALING ROUTINE
0906 A2 02              LDXIM $02    START PATTERN LOAD
0908 B5 0A      NOTEA   LDAZX $0A
090A 95 02              STAZX $02
090C CA                 DEX
090D 10 F9              BPL   NOTEA
090F 20 6F 08           JSR   TIMEC  TIMING ROUTINE
0912 A5 0D      NOTEB   LDA   $000D  REST PART OF NOTE
0914 F0 0E              BEQ   MAIN   IF NOTHING TO DO
0916 20 93 08           JSR   SCALE  SCALING ROUTINE
0919 A9 00              LDAIM $00
091B 85 02              STA   $0002  ZERO OUT PATTERN FOR
091D 85 03              STA   $0003  REST PART
091F 85 04              STA   $0004
0921 20 6F 08           JSR   TIMEC  TIMING

0924                    ORG   $0924

                MAIN PART OF INTERPRETER
                ENTRY AT "ENTRY"

0924 E6 13      MAIN    INC   $0013  TABLE ADDRESS
0926 D0 02              BNE   ENTRY
0928 E6 14              INC   $0014

092A A0 00      ENTRY   LDYIM $00
092C B1 13              LDAIY $0013  NEXT TABLE BYTE
092E 30 12              BMI   MAINA  TO CONTROL SECTION
0930 48                 PHA
0931 29 0F              ANDIM $0F    TONE
0933 0A                 ASLA
0934 A8                 TAY
0935 68                 PLA
0936 29 70              ANDIM $70    DURATION
0938 4A                 LSRA
0939 4A                 LSRA
093A 4A                 LSRA
093B 69 02              ADCIM $02    TOTAL DURATION IN 16THS
093D 85 0E              STA   $000E
093F 4C BE 08           JMP   NOTE   PAY NOTE

0942 C9 FD      MAINA   CMPIM $FD    C0 + 3D IS LONGEST NOTE FOR
0944 90 01              BCC   MAINB  FOR SCALING REASONS
0946 60                 RTS

0947 48         MAINB   PHA
0948 0A                 ASLA
0949 10 07              BPL   MAINC
094B 68                 PLA
094C 29 3F              ANDIM $3F    NOTE LENGTH
094E 85 12              STA   $0012
0950 B0 D2              BCS   MAIN   UNCONDITIONAL BRANCH
```

```
0952 0A        MAINC  ASLA
0953 10 08            BPL   MAIND
0955 68               PLA
0956 29 1F            ANDIM $1F     TONAL RANGE INDEX
0958 0A               ASLA
0959 85 0F            STA   $000F
095B 90 C7            BCC   MAIN    UNCONDITIONAL BRANCH

095D 0A        MAIND  ASLA
095E 10 07            BPL   MAINE
0960 68               PLA
0961 29 0F            ANDIM $0F     REST FRACTION
0963 85 0D            STA   $000D
0965 90 BD            BCC   MAIN    UNCONDITIONAL BRANCH

0967 0A        MAINE  ASLA
0968 10 03            BPL   MAING
096A 68        MAINF  PLA
096B 90 B7            BCC   MAIN    DUMMY, CONTROLS NOT INTERPRETED
096D 0A        MAING  ASLA
096E 30 FA            BMI   MAINF
0970 0A               ASLA
0971 10 2B            BPL   MAINI
0973 68               PLA
0974 AA               TAX           JSR AND JMP SECTION
0975 4A               LSRA
0976 90 0A            BCC   MAINH
0978 A5 13            LDA   $0013   JSR SECTION, PUSH RETURN TABLE
097A 69 01            ADCIM $01     ADDRESS ON TO STACK
097C 48               PHA
097D A5 14            LDA   $0014
097F 69 00            ADCIM $00
0981 48               PHA
0982 C8        MAINH  INY
0983 B1 13            LDAIY $0013   GET NEW ADDRESS
0985 48               PHA
0986 C8               INY
0987 B1 13            LDAIY $0013
0989 85 14            STA   $0014
098B 68               PLA
098C 85 13            STA   $0013
098E 8A               TXA           AND STORE IT FROM BEGINNING
098F 4A               LSRA          OF SELECTION
0990 90 98            BCC   ENTRY   JMP
0992 20 2A 09         JSR   ENTRY   JSR
0995 68               PLA
0996 85 14            STA   $0014   PULL ADDRESS AND STORE IT
0998 68               PLA
0999 85 13            STA   $0013
099B 18               CLC
099C 90 86            BCC   MAIN    UNCONDITIONAL BRANCH

099E 68        MAINI  PLA
099F A0 03            LDYIM $03     GET NEW PATTERN AND
09A1 B1 13     MAINJ  LDAIY $0013   STORE IT
```

```
09A3 99 09 00          STAY  $0009
09A6 88                DEY
09A7 D0 F8             BNE   MAINJ
09A9 A5 13             LDA   $0013
09AB 69 03             ADCIM $03      JUMP OVER PATTERN
09AD 85 13             STA   $0013
09AF 90 02             BCC   MAINK
09B1 E6 14             INC   $0014
09B3 4C 24 09   MAINK  JMP   MAIN

09C0                   ORG   $09C0

                INITIALIZATION FOR ZERO PAGE

09C0 D8         ZERO   CLD            JUST IN CASE
09C1 A9 00             LDAIM $00
09C3 85 10             STA   $0010
09C5 A9 08             LDAIM $08
09C7 85 11             STA   $0011
09C9 85 01             STA   $0001
09CB A9 0A             LDAIM $0A
09CD 85 14             STA   $0014    NOTE TABLE PAGE
09CF A9 20             LDAIM $20
09D1 85 13             STA   $0013    NTOE TABLE BYTE
09D3 A9 01             LDAIM $01
09D5 85 0D             STA   $000D    REST 16THS
09D7 A9 20             LDAIM $20
09D9 85 12             STA   $0012    NOTE LENGTH, CONTROLS TEMPO
09DB A9 20             LDAIM $20
09DD 85 0F             STA   $000F    TONAL RANGE INDEX
09DF A9 05             LDAIM $05
09E1 85 0A             STA   $000A    WAVE SHAPE PATTERN
09E3 85 0B             STA   $000B
09E5 85 0C             STA   $000C
09E7 20 2A 09          JSR   ENTRY    TO APPLAYER
09EA 4C 69 FF          JMP   $FF69    TO MONITOR, AFTER THE BEEP

09F8                   ORG   $09F8

                TABLE OF ENTRY POINTS FOR TIMING ROUTINE

09F8 63         TTABLE =     $63
09F9 6A                =     $6A
09FA 62                =     $62
09FB 6D                =     $6D
09FC 61                =     $61
09FD 6C                =     $6C
09FE 60                =     $60
09FF 6B                =     $6B

ENTRY  092A    MAIN   0924    MAINA  0942    MAINB  0947
MAINC  0952    MAIND  095D    MAINE  0967    MAINF  096A
MAING  096D    MAINH  0982    MAINI  099E    MAINJ  09A1
MAINK  09B3    NOTE   08BE    NOTEA  0908    NOTEB  0912
SCALE  0893    SCALEA 08B2    TIME   0860    TIMEA  0863
TIMEB  086A    TIMEC  086F    TIMED  0876    TIMEE  0881
TIMEF  088A    TIMEG  088E    TTABLE 09F8    ZERO   09C0
```

6502 BIBLIOGRAPHY
PART IV

William Dial
438 Roslyn Avenue
Akron, OH 44320

301. Michels, Richard E. "How to Buy an Apartment Building", Interface Age 3, No. 1, pp 94-99 (Jan 1978)
A 6502 FOCAL based system for handling the many factors involved via a computer decision making program.

302. Woods, Larry "How Are You Feeling Today?" Kilobaud No.14,pp24-30(Feb 1978)
Biorhythms with your KIM are displayed on the KIM readout.

303. Craig, John "Editor's Remarks" Kilobaud No. 14 p 22 (Feb 1978)
In a discussion of Microsoft Level II BASIC it is pointed out that Microsoft BASIC is being used on Altair 6800 and 8080, TRS-80, and 6502 based systems OSI, PET, KIM and Apple (floating-point version).

304. Bishop, Robert J. "Star Wars" Kilobaud No. 14 pp52-56 (Feb. 1978)
An Apple-II graphics game based on the 6502.

305. Blankenship, John "Expand Your KIM! Part 3" Kilobaud pp68-71 (Feb. 1978)
This installment covers bus control board and memory.

306. Burhans, R.W. "How Much Memory for a KIM?" Kilobaud p 118 (Feb. 1978)
Decoding the KIM for 28K.

307. Pearce, Craig A., p.6 suggestions for running graphics on the PET.
Julin, George, pp6-7, letter on PET graphics.
Stuck, H.L. p 7, more on the PET.
Above three are letters in Peoples Computers 6, No. 4, (Jan-Feb. 1978)

308. Wells, Edna H. "Program Abstract" Peoples Computers p 7 (Jan-Feb. 1978)
Program for the Commodore PET with 8K BASIC, entitled Graphics-to ASCII Utility--ASCIIGRAPH.

309. Cole, Phyllis "SPOT-The Society of PET Owners and Trainers", Peoples Computers No. 4, pp 16-19 (Jan-Feb 1978)
Notes for the Users of the PET.

310. Inman, Don "The Data Handler User's Manual: Conclusion" Peoples Computers No. 4 pp24-31 (Jan-Feb 1978)
The final installment of this series covers simple and inexpensive output devices.

311. Inman, Don "The First Book of KIM, Peoples Computers No. 4 p34 (Jan-Feb1978)
A good review of this excellent book.

312. Braun, Ludwig "Magic for Educators--Microcomputers" Personal Computing, 2 No. 1, pp 30-40 (Jan. 1978)
Discussion of micros includes the 6502 based Apple II and the PET.

313. Helmers, Carl "An Apple to Byte", BYTE 3, No. 3, p. 18-46 (Mar. 1978)
A user's reactions to the Apple II, with an example of a simple "color sketchboard" application.

314. Fylstra, Dan "User's Report: The PET 2001", BYTE,pp114-127 (Mar. 1978)
A fairly comprehensive report on the PET.

315. Brader, David, "KOMPUUTAR Updates", BYTE pp131-132 (Mar. 1978)
In a letter Brader responds to some inquiries on his KOMPUUTAR system based on 6502 which was published in BYTE, Nov. 1977.

316. Jennings, Peter R., "Microchess 1.5 versus Dark Horse", BYTE 3, No. 3 pp 166-167 (March 1978)
Microchess 1.5 is Jenning's new extended version of the original Microchess. It occupies 2.5K and runs on KIM-1 with expanded memory. It is still being developed but in a test game with Dark Horse, a 24K program written in Fortran IV, the new version did very well indeed.

317. Rindsberg, Don "Here's HUEY!...super calculator for the 6502", Kilobaud, No. 12, pp 94-99 (December 1977)
   The calculating power of FORTRAN with trig functions, natural and common logs, exponential functions, all in 2.5K.

318. Finkel, LeRoy "Every Home (School) Should Have a PET" Calculators/Computers, page 83 (October 1977)

319. Anon, "12-Test Benchmark Study Results Show How Microprocessors Stack Up (8080, 6800, 6502)", EDN, page 19 (November 20, 1977)

320. Gordon, H.T. "Decoding Efficiency and Speed", DDJ 3, Issue 2, No. 22, pp 5-7 (Feb., 1978)
   Pros and cons of table look-up in 650X microprocessors.

321. Green, Wayne "Publishers Remarks", Kilobaud, No. 16, p 4 (April 1978)
   In a column on microprocessors, Green indicates that MOS Technology has a SuperKIM being readied and also that books on expanding the KIM system are coming out.

322. Carpenter, Chuck "Letters: KIM-1, ACT-1; The Scene", Kilobaud p 18 (Apr 1978
   A generally favorable report of one user's experience in interfacing and using ACT-1 with the KIM-1.

323. Braun, Ludwig, "PET Problems", Personal Computing No. 3, pp 5-6 (March 1978)
   Some observations by a PET owner.

324. Lasher, Dana "The Exterminator--for Buggy KIMs" 73 Magazine (April, 1978)
   Hardware and Software for a debugging facility.

325. Eaton, John "Now Anyone Can Afford a Keyboard" 73 Magazine (April, 1978)
   A melding of a surplus keyboard, KIM and software.

326. Foster, Caxton C. "Programming a Microcomputer: 6502" Addison-Wesley Publishing Company, Reading, Mass. 1978
   Caxton C. Foster of the University of Massachusetts, Amherst, has put together a very helpful book on programming the 6502 using KIM-1 as a lab tool.

327. Barden, William, Jr. "Computer Corner - 6502" Radio-Electronics (May 1978)
   An in-depth look at the widely used 6502 microprocessor.

328. Wozniak, Steve "Renumbering and Appending Basic Programs on the Apple-II Computer" DDJ Issue 3 (March 1978)
   Comments and techniques for joining two BASIC programs into a single larger one.

329. Eaton, John "A KIM Binary Calculator" DDJ Issue 3 (March 1978)
   An easier way to solve binary math programs.

330. Wells, Ralph "PET's First Report Card" Kilobaud pp 22-30 (May 1978)
   An objective evaluation of PET serial No. 171.

331. Blankenship, John "Expand you KIM" Kilobaud, pp 60-63 (May 1978)
   Part 5: A/D interfacing for joysticks. Four channels.

332. Holland, Hugh C. "KIM Notes" BYTE 3 No. 4, p 163 (April 1978)
   Correction for Hal Chamberlin's Four Part Harmony Program published in September 1977 BYTE.

333. Anon., "Byte's Bits", BYTE 3, No. 4, p 166 (April 1978)
   Notes on picking the right color television for an Apple.

334. KIM-1 User Notes, Issue 9/10, (January - March 1978)
   - Rehnke, Eric "Have you been on the Bus" page 1.
   - Kushnier, Ron "Space War Phaser Sound" page 2.
   - Butterfield, Jim "Skeet Shoot" page 2.
   - Edwards, Lew "KIM D-BUG" page 3.
   - Flacco, Roy "Graphics Interface" page 4.
   - Wood, James "RPN Calculator Interface to KIM" page 6.
   - Bennett, Timothy "KUN Index by Subject, Issues 1 to 6" page 12.
   - Niessen, Ron "On Verifying Programs in RAM" page 12.
   - Pottinger, Hardy "Greeting Card Generator" page 13.

A BLOCK HEX DUMP AND CHARACTER MAP UTILITY PROGRAM FOR THE KIM-1

J. C. Williams
35 Greenbrook Drive
Cranbury, NJ 08512

Here's a useful, fully relocatable utility program which will dump a specified block of memory from a KIM to a terminal. At the user's option, a hex dump with an ASCII character map is produced.

The hex dump will allow the programmer to rapidly check memory contents against a "master" listing when loading or debugging programs. With a printing terminal, the hex dump produces documentation of machine code to complement an assembly listing of a program.

A character map is useful if the block being dumped is an ASCII file. An example would be source code being prepared with an editor for later assembly. The map shows what the file is and where it is in case a minor correction is needed using the KIM monitor.

To use this utility program:

1. Load the code anywhere you want it, in RAM or PROM memory.

2. Define the block to be dumped just as for a KIM-1 tape dump:

   BLOCK STARTING ADDRESS 17F5 (low)
   17F6 (high)
   BLOCK ENDING ADDRESS+1 17F7 (low)
   17F8 (high)

3. Select the MAP/NOMAP option:

   MAP mode 00 in 17F9
   NOMAP mode FF in 17F9

4. Run the program starting at the first instruction. At the end of the dump, control will return to the KIM monitor. The examples following the assembly listing will give you the idea.

The program as listed dumps 16 decimal bytes per line. Users with TVT's may want to initialize the line byte counter for 8 decimal bytes per line to allow the hex with MAP format to fit the display. To make this change, replace the $0F at $021E with $07.

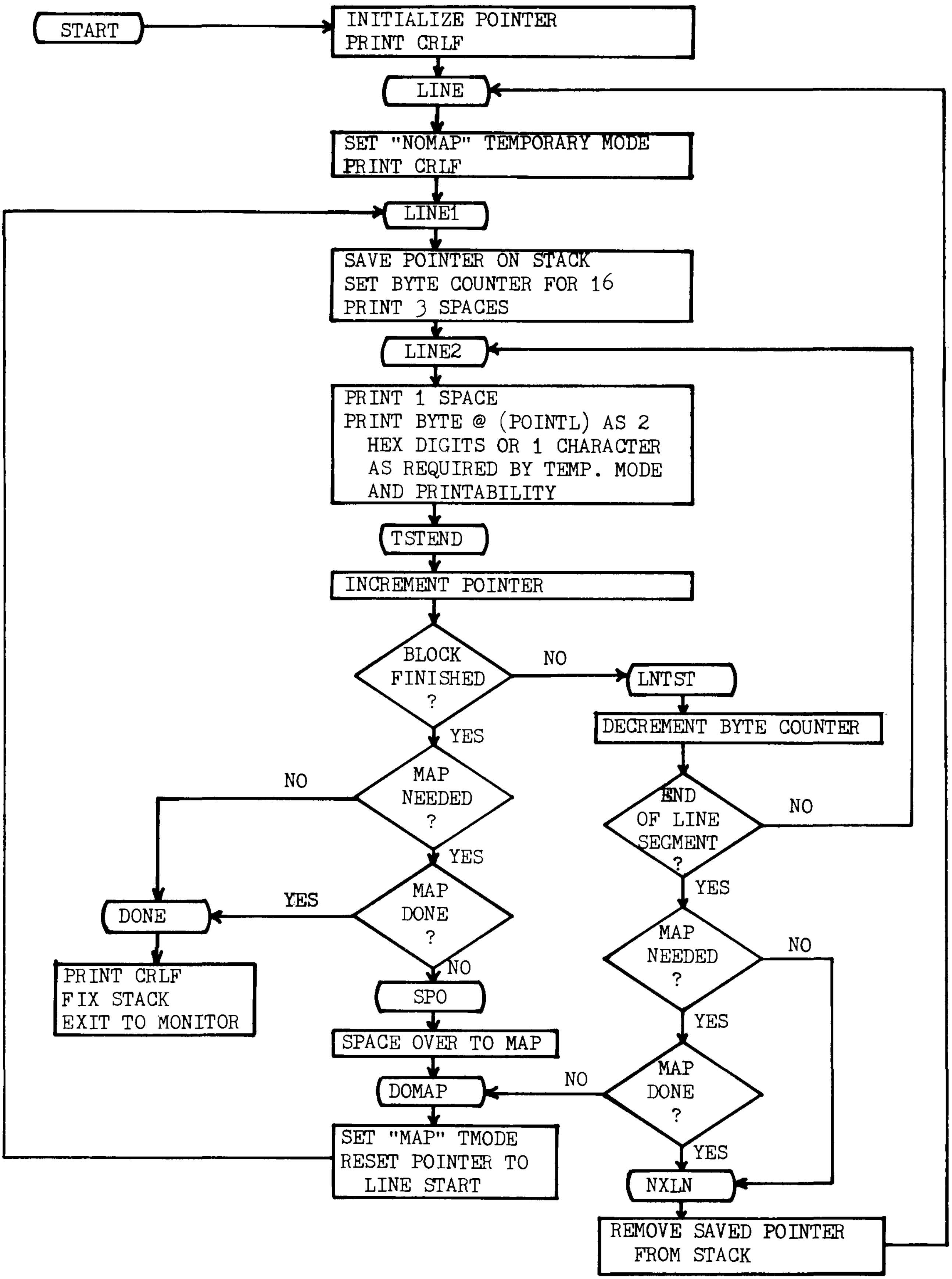
Another possible change is to have the program exit to a location other than the KIM-1 monitor. Exit to a text editor or tape dump may be convenient. Since the MAP/NOMAP option is determined by the most significant (sign) bit of what is stored at $17F9, a suitable tape ID number can be placed there for use of the KIM-1 tape dump or Hypertape. Use ID's from $01-$7F for files needing no character map and ID's from $80-$FE for ASCII files. Start the tape recorder in RECORD when the dump to the terminal is a few seconds from completion.

The flowchart will assist users wanting to make major alterations. Of necessity, ASCII non-printable characters are mapped as two hex digits. If other ASCII codes have special meaning for the user's terminal, a patch will be necessary to trap them. Single-stepping through this program can't be done because it uses the monitor's "display" locations. This is a small price to pay in order to use the monitor's subroutines. If use with a non-KIM 650X system is desired, the subroutines used must preserve the X register.

SYMBOL TABLE

CRLF	1E2F	DOMAP	026E	DONE	028A	EAH	17F8
EAL	17F7	EXT	1C4F	INCPT	1F63	INIT	0200
LINE	020D	LINEA	0217	LINEB	0228	LNTST	0279
MODE	17F9	NXLN	0285	OUTCH	1EA0	OUTSP	1E9E
POINTH	00FB	POINTL	00FA	PRTBYT	1E3B	PRTPNT	1E1E
PTBT	0243	SAH	17F6	SAL	17F5	SPO	0262
TMODE	00F9	TSTEND	0247				

BLOCK HEX DUMP WITH CHARACTER MAP



```
                    BLOCK HEX DUMP AND CHARACTER MAP
                    UTILITY PROGRAM FOR KIM-1

                    J. C. WILLIAMS - 1978

0200                       ORG   $0200

                    MEMORY LOCATIONS

0200                TMODE  *     $00F9  TEMPORARY MODE FLAG
0200                POINTL *     $00FA  POINTER
0200                POINTH *     $00FB
0200                SAL    *     $17F5  BLOCK STARTING ADDRESS
0200                SAH    *     $17F6
0200                EAL    *     $17F7  BLOCK ENDING ADDRESS + 1
0200                EAH    *     $17F8
0200                MODE   *     $17F9  00 FOR NO MAP, FF FOR HEX AND MAP
0200                EXT    *     $1C4F  EXIT TO KIM MONITOR

                    SUBROUTINES IN KIM MONITOR

0200                OUTCH  *     $1EA0  PRINTS BYTE IN A AS ONE ASCII CHARACTER
0200                CRLF   *     $1E2F  CARRIAGE RETURN AND LINE FEED
0200                OUTSP  *     $1E9E  PRINTS ONE SPACE
0200                PRTBYT *     $1E3B  PRINTS BYTE IN A AS TWO HEX DIGITS
0200                PRTPNT *     $1E1E  PRINTS POINTER
0200                INCPT  *     $1F63  INCREMENTS POINTER

0200 AD F5 17       INIT   LDA   SAL    INITIALIZE POINTER
0203 85 FA                 STA   POINTL
0205 AD F6 17              LDA   SAH
0208 85 FB                 STA   POINTH
020A 20 2F 1E              JSR   CRLF

020D A9 00          LINE   LDAIM $00    START A LINE
020F 85 F9                 STA   TMODE  INTI TMODE
0211 20 2F 1E              JSR   CRLF
0214 20 1E 1E              JSR   PRTPNT PRINT POINTER
0217 A5 FA          LINEA  LDA   POINTL START A LINE SEGMENT
0219 48                    PHA
021A A5 FB                 LDA   POINTH
021C 48                    PHA
021D A2 0F                 LDXIM $0F    INIT BYTE COUNTER
021F 20 9E 1E              JSR   OUTSP  OUTPUT SOME SPACES
0222 20 9E 1E              JSR   OUTSP
0225 20 9E 1E              JSR   OUTSP
0228 20 9E 1E       LINEB  JSR   OUTSP
022B A0 00                 LDYIM $00    GET THE BYTE
022D B1 FA                 LDAIY POINTL AND SAME ON STACK
022F 48                    PHA
0230 24 F9                 BIT   TMODE  IN MAP MODE?
0232 10 0F                 BPL   PTBT   NO
0234 29 7F                 ANDIM $7F    YES.  TEST FOR PRINTABLE
0236 C9 20                 CMPIM $20    CHARACTER
0238 30 09                 BMI   PTBT   PRINT AS TWO HEX DIGITS
023A 68                    PLA
```

```
023B 20 A0 1E          JSR   OUTCH  PRINT AS ONE ASCII CHARACTER
023E 20 9E 1E          JSR   OUTSP  AND A SPACE
0241 10 04             BPL   TSTEND UNCONDITIONAL BRANCH
0243 68       PTBT     PLA          RECOVER BYTE AND
0244 20 3B 1E          JSR   PRTBYT PRINT AS TWO HEX DIGITS

0247 20 63 1F TSTEND   JSR   INCPT  INCREMENT POINTER
024A A5 FA             LDA   POINTL AND TEST AGAINST ENDING
024C CD F7 17          CMP   EAL    ADDRESS + 1
024F A5 FB             LDA   POINTH
0251 ED F8 17          SBC   EAH
0254 90 23             BCC   LNTST  NOT BLOCK END.  TEST FOR LINE END
0256 2C F9 17          BIT   MODE   END OF BLOCK REACHED.  IS MAP
0259 10 2F             BPL   DONE   NEEDED.  DONE IF NOT.
025B 24 F9             BIT   TMODE  HAS MAP BEEN DONE?
025D 30 2B             BMI   DONE   YES, EXIT
025F CA                DEX
0260 30 0C             BMI   DOMAP  NO SPACES NEEDED
0262 20 9E 1E SPO      JSR   OUTSP  SPACE OVER TO CHARACTER MAP
0265 20 9E 1E          JSR   OUTSP
0268 20 9E 1E          JSR   OUTSP
026B CA                DEX
026C 10 F4             BPL   SPO

026E C6 F9    DOMAP    DEC   TMODE  DO THE MAP.  FIRST SET THE
0270 68                PLA          MAP FLAG AND RESET POINTER TO
0271 85 FB             STA   POINTH START OF LINE
0273 68                PLA
0274 85 FA             STA   POINTL
0276 38                SEC
0277 B0 9E             BCS   LINEA  AND PRINT THE MAP SEGMENT

0279 CA       LNTST    DEX          TEST FOR END OF LINE
027A 10 AC             BPL   LINEB  NOT AT END.  DO THE NEXT BYTE
027C 2C F9 17          BIT   MODE   END OF LINE SEGMENT REACHED.  IS MAP NEEDED?
027F 10 04             BPL   NXLN   NO, DO THE NEXT LINE
0281 24 F9             BIT   TMODE  HAS THE MAP SEGMENT BEEN DONE?
0283 10 E9             BPL   DOMAP  NO, DO IT NOW

0285 68       NXLN     PLA          DO THE NEXT LINE
0286 68                PLA          FIRST FIXT THE STACK
0287 38                SEC
0288 B0 83             BCS   LINE   DO THE NEXT LINE

028A 20 2F 1E DONE     JSR   CRLF   DONE
028D 68                PLA          REMOVE SAVED POINTER FORM STACK
028E 68                PLA
028F 4C 4F 1C          JMP   EXT    EXIT TO KIM MONITOR
```

```
KIM
2880 52 17F5
17F5 00 00.   BLOCK STARTING ADDRESS = 2800
17F6 28 28.
17F7 80 80.   BLOCK ENDING ADDRESS + 1 = 2880
17F8 28 28.
17F9 00 FF.   SELECT MAP OPTION
17FA FF 021E
021E 0F 07.   SELECT 8 LOCATIONS PER LINE
021F 20 0200
0200 AD G     START PROGRAM AT 0200

2800      0D 00 10 20 20 20 42 4C    0D  00  10              B   L
2808      4F 43 4B 20 48 45 58 20    O   C   K       H   E   X
2810      44 55 4D 50 20 41 4E 44    D   U   M   P       A   N   D
2818      20 43 48 41 52 41 43 54        C   H   A   R   A   C   T
2820      45 52 20 4D 41 50 0D 00    E   R       M   A   P   0D  00
2828      20 20 20 20 55 54 49 4C                    U   T   I   L
2830      49 54 59 20 50 52 4F 47    I   T   Y       P   R   O   G
2838      52 41 4D 20 46 4F 52 20    R   A   M       F   O   R
2840      4B 49 4D 2D 31 0D 00 30    K   I   M   -   1   0D  00  0
2848      0D 00 40 20 20 20 4A 2E    0D  00  @               J   .
2850      20 43 2E 20 57 49 4C 4C        C   .       W   I   L   L
2858      49 41 4D 53 20 2D 20 31    I   A   M   S       -       1
2860      39 37 38 0D 00 50 0D 00    9   7   8   0D  00  P   0D  00
2868      60 20 4F 52 47 20 24 30    `       O   R   G       $   0
2870      32 30 30 0D 00 70 0D 00    2   0   0   0D  00  p   0D  00
2878      80 20 20 20 4D 45 4D 4F    80              M   E   M   O
```

```
KIM
17F5
17F5 00 00.   BLOCK STARTING ADDRESS = 2800
17F6 28 28.
17F7 80 80.   BLOCK ENDING ADDRESS + 1 = 2880
17F8 28 28.
17F9 FF 00.   SELECT NOMAP OPTION
17FA FF 021E
021E 07 0F.   SELECT 16 LOCATIONS PER LINE
021F 20 0200
0200 AD G     START PROGRAM AT 0200

2800      0D 00 10 20 20 20 42 4C 4F 43 4B 20 48 45 58 20
2810      44 55 4D 50 20 41 4E 44 20 43 48 41 52 41 43 54
2820      45 52 20 4D 41 50 0D 00 20 20 20 20 55 54 49 4C
2830      49 54 59 20 50 52 4F 47 52 41 4D 20 46 4F 52 20
2840      4B 49 4D 2D 31 0D 00 30 0D 00 40 20 20 20 4A 2E
2850      20 43 2E 20 57 49 4C 4C 49 41 4D 53 20 2D 20 31
2860      39 37 38 0D 00 50 0D 00 60 20 4F 52 47 20 24 30
2870      32 30 30 0D 00 70 0D 00 80 20 20 20 4D 45 4D 4F
```

APPLE II ACCESSORIES AND SOFTWARE

Chuck Carpenter W5USJ
2228 Montclair Place
Carrollton, TX 75006

Apple II owners may find a couple of new items as interesting as I did.

First, a renumber and append machine language program. This was published in Dr. Dobbs, Issue #23, April 1978. Renumber lets you change line numbers on your entire program or any part of it. It renumbers branching statements too. Append lets you link two programs together. Any program you have in the machine needs to have higher line numbers than the one being loaded from tape. Renumber lets you do this. POKE commands load the various starting and ending addresses. CALL commands execute the renumber or append program. Caution: Renumber and Append will work only with integer BASIC.

Second, the serial interface board from Electronic Systems, San Jose, CA. They are definitely among the "Good Guys". I ordered the parts on a Thursday (by phone) and received them the following Monday. That's what I call rapid response. I ordered the serial board assembled and the TTL to RS232 board and the MODEM board as kits. I don't have the latter two built yet, but I intend to have communicating ability when I get done. Workmanship and quality on the assembled board and the kits was satisfactory (and I'm fussy). The serial board instructions are a bit vague. Unless you are quite familiar with the Apple's monitor, BASIC and various I/O port commands and addresses, you are likely to have some problems. Also, I couldn't make the terminal program work and there was no explanation of what it was supposed to do.

However, the price is attractive ($62 assembled and tested, $42 kit) and the service was great. I expect eventually that I'll be able to have an inexpensive communicating terminal. The MODEM board can be originate or answer so I'll have to use two if I want to do both. A note of caution here too. As written, the machine language program starts at page 3 ($0300). Applesoft BASIC uses the first few bytes of this page. You'll have to relocate the terminal part of the program to use both integer and floating point BASIC. I have the serial board connected to my printer and everything works okay. I'll pass along the results when I have the system set up to communicate.

Finally, Apple has a new version of Applesoft called Applesoft II. This became available in April 1978. The new version is 1.5K longer but has all the standard integer BASIC commands and a few more. It is not compatible with previous versions of Applesoft. All the known problems seem to have been corrected. It's really nice to be able to go from one BASIC to the other and have to remember only the extended capabilities, especially for LORES graphics. There are commands to FLASH and RESTORE screen characters, a SPEED command to vary the screen writing rate, and you can develop HIRES graphics directly from program control. Maybe we Apple owners should request a retrofit kit. This way we can catch up on all of the new goodies that are coming from Apple. Especially the documentation.

Addendum - by Robert M. Tripp

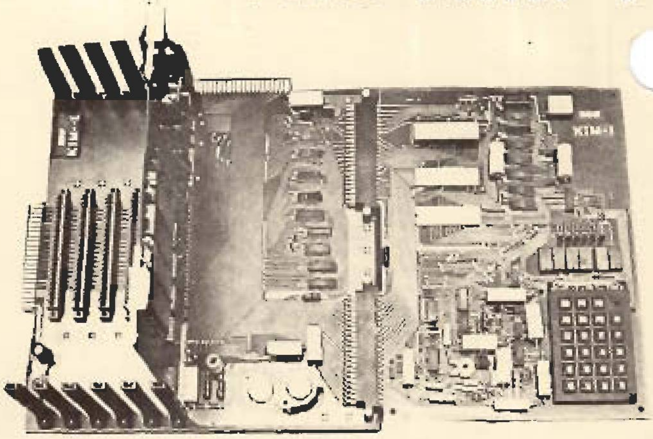
Speaking of documentation, I was quite pleased to receive the "Apple II BASIC Programming Manual" by Jef Raskin, Published by Apple Computer Company, 1978. This arrived in the mail, unsolicited. I assume that all Apple II owners have received one. If not, write Apple and ask for it: product #A2L0005X. The manual is well written and elegantly printed. My only minor complaint is that the light green ink used to show the display contents make the book a little difficult to read. I hope that this manual is only the first of many that we will be seeing from Apple. It is a very good start.

the Computer Store

63 SOUTH MAIN STREET, WINDSOR LOCKS, CONNECTICUT 06096
203-627-0188

The Computer Store is pleased to announce off-the-shelf availability of **Apple II™, the personal computer.**

The KIM 1



the Computer Store

63 SOUTH MAIN STREET • WINDSOR LOCKS, CONNECTICUT

GIFT CERTIFICATE

THIS COUPON GOOD FOR $2 OFF ANY PURCHASE OVER $5.