6502 USER NOTES

NO. 14 $2.50

EDITORIAL

From the feedback I've received concerning issue #13, most of you were very happy about the "new" format so things will stay pretty much the same. A good number of comments specifically mentioned their satisfaction with the "language lab" so we may expand on it a bit in future issues. Lots of good stuff in store for you BASIC, FOCAL and TINY BASIC users. Thanks to all of you who responded to my questions on modifying BASIC to skip the initialization messages. As you will soon know, that question and many others will be answered in this issues' BASIC column. Also, thanks to Dick Grabowski of HDE and Bob Kurtz of Micro-Z, we now have a new command for BASIC.

A number of you have been asking for some terminal-oriented software so we have modified the original APPLE disassembler for the KIM. You'll need more memory, but if you have a terminal you're finding that out anyway.

Those of you with hard copy will no doubt enjoy the BANNER program. The present character set is designed for a 40 column printer but can easily (?) be re-designed for wider terminal widths. When you do come up with a new character set, send it in so the rest of us can enjoy it. Send it in on a cassette to make life easier for us and we'll publish it and/or offer it on cassette, depending on its design. There's enough info in the article to enable you to design your own character set and further info can be gotten from the Kilobaud article.

SUBMITTING ARTICLES

Since all articles will be retyped they need only be readable. Typing it would, of course, guarantee readability. Program listings, on the other hand, may not be retyped so, if at all possible, use white paper and a fresh ribbon on your printer. If there's no way you can generate an original source listing, then a handwritten source listing with MOS mnemonics, and labels of up to six characters, (don't forget to use labels when referencing zero page locations) will be satisfactory. Comments should be preceded by a semicolon.

This will make it easy for me to assemble your program for publication. Disassembler output is not very satisfactory except when heavily commented, labeled and all zero page registers identified by name.

Perhaps the best way to submit program source listings would be to send a cassette of the assembler source file and I can then assemble it and run a listing on my Decwriter. I Can assemble source files from either the Micro-ade assembler (Peter Jennings) or the MOS/ARESCO/HDE assemblers. If you send a S.A.S.E.,I'll return your cassettes. It would be wise to dump two copies of the file to cassette just in case.

I can read most of the Hypertape-recorded cassettes I receive once I adjust the azimuth of the cassette head for the highest audio level while reading the program. I think this head adjustment problem has probably accounted for most of the tape interchange problems I've been aware of. The machines I use to make the newsletter cassettes have been adjusted as close as possible and 30 seconds of synch characters precede the program for setting up your equipment. So far, we have not had any cassettes returned, so we must be doing something right.

6502 USER NOTES is published bimonthly by Eric C. Rehnke (POB 33093, N. Royalton, Ohio 44133 (216-237-0755)). Subscription rates for Volume 3 are $13.00 (US and Canada) and $19.00 elsewhere. No part of 6502 USER NOTES may be copied for commercial purposes without the express written permission of the publisher. Articles herein may be reprinted by club newsletters as long as proper publication credit is given and the publisher is provided with a copy of the publication.

BACK ISSUE AVAILABILITY

"Official" reprints of USER NOTES 1-12 are now available from Mark Kantrowitz, 15 Midway Ct., Rockaway NJ 07866. Prices are $5.00 for issues 1-6 or 7-12 (1st Class in N America)- $10.00 for issues 1-12 (1st Class in N America)- $13.00 for issues 1-12 (overseas) U.S. Funds only.

LET THE BUYER BEWARE!!

Those of you who have been around this industry for awhile know by now that just 'cuz somethings advertised doesn't mean that it really exists.

I am very interested in hearing about your experiences with any of the advertisers in USER NOTES.

When purchasing hardware or software, it makes alot of sense to purchase the documentation ahead of time to see what you're getting into. The quality of documentation can, prove to be a good indicator of the company's performance in other areas pertaining to that product.

On the other hand, do be reasonable. Don't expect a 60 page manual to accompany a $5 or $10 software package. I'm really referring to products of medium or high complexity such as some Assemblers or high-level languages, floppy-disc drives, prom programmers, video boards etc.

For instance, are there detailed instructions for getting the product running on your system? Does the product need some non-standard hardware or software? Are there enough examples to make operation fairly straightforward? What if you have problems? Are there some trouble shooting hints in the manual? How 'bout a phone number to call if you have problems which you can't handle? For items which may cost from several hundred to several thousand dollars, it would be a good idea to call the company with some real or made-up questions just to see what kind of response you get.

Are you treated courteously? Do you get connected to someone who can answer your question? (If the right person doesn't happen to be in the office when you call, don't expect them to return a long distance call.)

It can get pretty lonesome out there when you've got a product that isn't performing and a company that ignores you. It's better to find out in advance how a company treats its customers. If something breaks, how will they back you up?

When you shop around for something big (like a floppy, for example) you should understand that price alone should not be THE determining factor. Other things to consider include: What kind of software comes with it? Will it interface easily with the particular high-level language I will want to use? What kinds of software can be used with it-any optional packages from the manufacturer? Can I interface the floppy system easily to some non-standard hardware I may want to add? How easy will it be to incorporate some improved software which gets released at a later date from the manufacturer?

(RAM based software that is brought in by means of a simple bootstrap program is generally much easier to upgrade than a ROM based operating system). Can the system software be backed-up on an extra disc? What will the manufacturer be offering in the next year or so? Do his plans sound reasonable? What else has he done? Past performance generally indicates future performance. If you're looking for the "cheapest" product-keep in mind that the manufacturer of the "cheapest" product probably can't afford to support his product next year. INVESTIGATE FULLY BEFORE YOU BUY!!!!!!

KIMSI USERS

Apparently Forethought Products, makers of KIMSI, have given up any plans to put out a newsletter of KIMSI information. I know there are alot of you out there so how 'bout if we have a section of USER NOTES just for you?

SOFTWARE FEATURES

KIM-1 BANNER

Jim Zuber
20224 Cohasset #16
Canoga Park, Ca 91306

If your KIM, SYM, or AIM system is hooked up to a printer or teletype get ready to have some fun! In the January 1979 issue of Kilobaud there was an article on page 64 called "Say it with Banner". This program prints out giant characters on your printer. There were three problems with the program:

1. It is written in 8080 code.
2. It uses octal notation.
3. It uses almost 8K of memory.

I took the general concept of printing large characters and wrote an orginal program that has the following features:

1. Written in 6502 code.
2. I/O independent.
3. Uses HEX notation.
4. Only uses 2K of memory.
5. Relocatable data tables.
6. Easy user modification of character sets.

Let's talk about the I/O configuration first. Location 2004 and 2005 defines the Input character routine location for your system. If your terminal echos your input change 2003 to 4C. The character output routine location for your system is defined by locations 2007 and 2008. Your output routine must do the following:

1. Provide a line feed if necessary.
2. Provide null characters if necessary.
3. Must preserve the X, Y, and accumulator registers.

Text can be stored anywhere in memory and is defined by locations 200C and 200D. The text string can be as long as you want as long as you don't run out of memory. The data tables can be stored anywhere in memory as long as the starting address of the tables is stored in 2009 and 200A. SYM users will want to store the tables right after the program. The print character is defined in location 200B. Use the HEX equivalant of the ASCII character you want. The program is set up to use an @. I will explain later on how to make up your own characters or modify some of the ones I made up. To use the program start at 2000 and GO. You will see a prompt (>>). Type in the text you want printed out. The program treats a carriage return as a space so take note. You terminate the text input with an @. If you typed in all valid characters you will see "@ o k" printed. Get your paper ready and type a carriage return to start the printing. If you type an illegal character you will see "@ " with the illegal characters sandwiched between the @ and the . Retype the text using only legal characters. At the end of the printout the program will prompt for more text. The legal characters are A thru Z, 0 thru 9, space, c/r, and the following characters: * . - + : ! ; ? $, . My characters are 10 rows by 35 columns. Obviously this is too big for the AIM printer. Don't worry, you can make up your own character set to work on the AIM. To create your own character set just follow these simple rules:

1. Always store ff at the end of the tables.
2. The first BYTE should be the HEX equivalent of the ASCII character.
3. The second BYTE should be the HEX number of data Bytes.
4. Carriage returns are defined by EE.
5. Store the configuration of the character in a serial manner.
6. A "print spaces" data Byte is defined by Bit 7 being set to zero and Bits 0 thru 6 set to the number of spaces you want printed. Example: 07 would print 7 spaces.
7. To print a mark (or a character) set Bit 7 to one and Bits 0 thru 6 set to the number of marks. Example: 87 would print 7 @'s.

Maybe this will help you understand a little better. In order to print an "1" (one) that is 15 columns by 7 rows wide, just put this in the tables: 31 0A EE EE 8F EE 8F EE 8F EE EE EE. The 31 is the HEX equivalent of ASCII character one. The 0A is the number of data Bytes. Then I print 2 carriage returns, 3 rows of 15 characters and 2 more carriage returns. Hope you enjoy this program. If you want to modify any of my characters you can find their location by storing the character in 0004, then call the find character subroutine. The character's location plus 2 will be stored in 0000 and 0001.

```
2000                    ;KIM-1 BANNER PROGRAM
2000                    ;WRITTEN BY JIM ZUBER 12/23/78
2000
2000                          *=$0
0000                    PNTL  *=*+1
0001                    PNTH  *=*+1
0002                    BUF1  *=*+1
0003                    BUF2  *=*+1
0004                    TEMP  *=*+1
0005                    TEMPX *=*+1
0006                    TEMPY *=*+1
0007
0007                    ;KIM I/O
0007                    GETCH =$1E5A
0007                    OUTCH =$1EA0
0007                    CRLF  =$1E2F
0007
0007                    EOS   =$40              ;END OF STRING CHAR
0007
0007                          *=$2000
2000  4C 0E 20          STAR  JMP OVER
2003  4C 2C 21          INV   JMP INPT          ;INPUT ROUTINE
2006  4C 34 21          OUTV  JMP OUTC          ;OUT VECTOR
2009  00                TBLL  .BYTE $00         ;TABLE LOW
200A  30                TBLH  .BYTE $~~30~~ 22      ;TABLE HIGH
200B  40                PRCH  .BYTE $40         ;PRINT CHAR
200C  00                BUFL  .BYTE $~~00~~ 50      ;BUFFER LOW
200D  40                BUFH  .BYTE $~~40~~ 21      ;BUFFER HIGH
200E
200E  D8                OVER  CLD
200F  A0 00                   LDY #0
2011  20 FA 20                JSR INTB          ;INPUT TEXT
2014  A9 3E                   LDA #'>           ;PROMPT CHAR
2016  20 06 20                JSR OUTV
2019  20 06 20                JSR OUTV
201C  A9 0D                   LDA #$0D          ;SEND A CR
201E  20 06 20                JSR OUTV
```

```
@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@        @@@@@@@@@@@@@@@@@@@@@@
@@@@@        @@@@@
@@@@@        @@@@@
@@@@@        @@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@



@@@@@                         @@@@@
@@@@@                         @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@



@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                          @@@@@@@@@
                    @@@@@@@@@
             @@@@@@@@@
             @@@@@@@@@
      @@@@@@@@@@
@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@



@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                          @@@@@@@@@
                    @@@@@@@@@
             @@@@@@@@@
             @@@@@@@@@
      @@@@@@@@@@
@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@



@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@          @@@@@
@@@@@          @@@@@
@@@@@          @@@@@
@@@@@          @@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@



    @@@@@@@@@@@     @@@@@@@@@@@
   @@@@@@@@@@@@@   @@@@@@@@@@@@@
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@          @@@@@          @@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
2021  20 03 20   CHAR   JSR INV          ;INPUT STRING
2024  91 02             STA (BUF1),Y
2026  C9 40             CMP #EOS         ;END OF STRING?
2028  F0 06             BEQ CHEK
202A  20 10 21          JSR INCB
202D  4C 21 20          JMP CHAR
2030  20 FA 20   CHEK   JSR INTB         ;CHECK CHARS
2033  A0 00      LOP3   LDY #0
2035  B1 02             LDA (BUF1),Y
2037  C9 40             CMP #EOS         ;END STRING?
2039  F0 17             BEQ OK
203B  85 04             STA TEMP
203D  20 C9 20          JSR FDCH         ;FIND CHAR
2040  C9 FF             CMP #$FF         ;IS IT BAD?
2042  D0 08             BNE LOP4
2044  A5 04             LDA TEMP

2046  20 06 20          JSR OUTV         ;START OVER
2049  4C 00 20          JMP STAR
204C  20 10 21   LOP4   JSR INCB
204F  4C 33 20          JMP LOP3
2052  A9 4F      OK     LDA #'O          ;PROMPT "OK"
2054  20 06 20          JSR OUTV
2057  A9 4B             LDA #'K
2059  20 06 20          JSR OUTV
205C  A9 0D             LDA #$0D
205E  20 06 20          JSR OUTV
2061  20 03 20          JSR INV          ;WAIT FOR KEY
2064  20 FA 20          JSR INTB         ;READY TO PRINT
2067  A0 00      LOP6   LDY #0
2069  B1 02             LDA (BUF1),Y
206B  C9 40             CMP #EOS         ;END?
206D  D0 03             BNE LOP7
206F  4C 00 20          JMP STAR
2072  85 04      LOP7   STA TEMP
2074  20 C9 20          JSR FDCH         ;FIND CHAR
2077  20 8B 20          JSR PNTC         ;PRINT IT
207A  A9 0D             LDA #$0D         ;3 ROWS
207C  20 06 20          JSR OUTV
207F  20 06 20          JSR OUTV
2082  20 06 20          JSR OUTV
2085  20 10 21          JSR INCB         ;INC BUFFER
2088  4C 67 20          JMP LOP6
208B
208B  A0 00      PNTC   LDY #0           ;PRINT CHAR
208D  B1 00             LDA (PNTL),Y     ;SUBROUTINE
208F  C9 EE             CMP #$EE         ;TIME TO CARRIAGE RETURN?
2091  D0 0B             BNE LP10
2093  A9 0D             LDA #$0D         ;OUTPUT C/R
2095  20 06 20          JSR OUTV
2098  4C BF 20          JMP STOP
209B  85 04      LP10   STA TEMP         ;GET DATA
209D  29 80             AND #$80         ;MARK OR SPACE
209F  D0 0D             BNE MARK
20A1  A4 04             LDY TEMP         ;MUST BE SPACE
20A3  A9 20      LP11   LDA #$20         ;OUTPUT SPACE
20A5  20 06 20          JSR OUTV
20A8  88                DEY
20A9  F0 14             BEQ STOP         ;ANY MORE?
20AB  4C A3 20          JMP LP11         ;MUST BE
20AE  A5 04      MARK   LDA TEMP         ;MUST BE MARK
20B0  29 7F             AND #$7F         ;MASK BIT 7
20B2  A8                TAY
20B3  AD 0B 20   LP12   LDA PRCH GET PRINT CHAR
20B6  20 06 20          JSR OUTV         ;OUTPUT MARK
20B9  88                DEY
20BA  F0 03             BEQ STOP         ;ANY MORE?
20BC  4C B3 20          JMP LP12         ;MUST BE
20BF  CA         STOP   DEX              ;CHECK END
20C0  F0 06             BEQ LP13

20C2  20 1E 21          JSR INCP         ;INC POINTER
20C5  4C 8B 20          JMP PNTC         ;GO BACK
20C8  60         LP13   RTS
20C9
20C9  20 05 21   FDCH   JSR INTP         ;FIND CHARACTER
20CC  A0 00      LOP1   LDY #0           ;SUBROUITNE
20CE  B1 00             LDA (PNTL),Y     ;PICK UP ILLEGAL?
20D0  C9 FF             CMP #$FF
20D2  F0 1B             BEQ OUT
20D4  C5 04             CMP TEMP         ;RIGHT ONE?
20D6  F0 18             BEQ OUT1
20D8  C8                INY              ;MUST NOT BE
20D9  B1 00             LDA (PNTL),Y     ;BYTE COUNT
20DB  18                CLC
20DC  65 00             ADC PNTL         ;ADD TO POINTER
20DE  85 00             STA PNTL
20E0  A9 00             LDA #0
20E2  65 01             ADC PNTH
20E4  85 01             STA PNTH
20E6  20 1E 21          JSR INCP
20E9  20 1E 21          JSR INCP
20EC  4C CC 20          JMP LOP1         ;LOOK AGAIN
20EF  60         OUT    RTS
```

```
20F0  20 1E 21   OUT1   JSR INCP        ;LOOK AT DATA
20F3  B1 00             LDA (PNTL),Y
20F5  AA                TAX             ;BYTE IN X
20F6  20 1E 21          JSR INCP
20F9  60                RTS
20FA
20FA  AD 0C 20   INTB   LDA BUFL        ;INITIALIZE
20FD  85 02             STA BUF1        ;BUFFER SUB
20FF  AD 0D 20          LDA BUFH
2102  85 03             STA BUF2
2104  60                RTS
2105
2105  AD 09 20   INTP   LDA TBLL        ;INITIALIZE
2108  85 00             STA PNTL        ;POINTER SUB
210A  AD 0A 20          LDA TBLH
210D  85 01             STA PNTH
210F  60                RTS
2110
2110  18         INCB   CLC             ;INCREMENT
2111  A5 02             LDA BUF1        ;BUFFER SUD
2113  69 01             ADC #$1
2115  85 02             STA BUF1
2117  A5 03             LDA BUF2
2119  69 00             ADC #0
211B  85 03             STA BUF2
211D  60                RTS
211E
211E  18         INCP   CLC             ;INCREMENT
211F  A5 00             LDA PNTL
2121  69 01             ADC #1          ;POINTER SUB
2123  85 00             STA PNTL
2125  A5 01             LDA PNTH
2127  69 00             ADC #0
2129  85 01             STA PNTH
212B  60                RTS
212C
212C  84 06      INPT   STY TEMPY       ;SAVE Y
212E  20 5A 1E          JSR GETCH       ;GET A CHAR
2131  A4 06             LDY TEMPY
2133  60                RTS
2134
2134  48         OUTC   PHA             ;SAVE CHAR
2135  86 05             STX TEMPX       ;AND X AND Y
2137  84 06             STY TEMPY
2139  C9 0D             CMP #$0D        ;IS IT A C/R?
213B  D0 06             BNE CONT
213D  20 2F 1E          JSR CRLF
2140  4C 46 21          JMP RESTOR      ;GET BACK THE REGS AND RETURN
2143  20 A0 1E   CONT   JSR OUTCH       ;OTHERWISE USE KIM OUTPUT.
2146  A6 05      RESTOR LDX TEMPX
2148  A4 06             LDY TEMPY
214A  68                PLA             ;RETORE THE ACC.
214B  60                RTS             ;AND RETURN
214C             FINISH .END
```

```
2200  20 07 EE EE EE EE EE EE EE 41 20 A3 EE A3 EE A3    2400  84 EE 84 1B 84 EE A3 EE A3 EE 50 23 A3 EE A3 EE
2210  EE 0F 85 0A 85 EE 0F 85 0A 85 EE 0F 85 0A 85 EE    2410  A3 EE 11 85 08 85 EE 11 85 08 85 EE 11 85 08 85
2220  0F 85 0A 85 EE A3 EE A3 EE A3 EE 42 2B A3 EE A3    2420  EE 11 85 08 85 EE 11 92 EE 11 92 EE 11 92 EE 51
2230  EE A3 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE 85    2430  20 A3 EE A3 EE 84 1B 84 EE 84 1B 84 EE 84 1B 84
2240  0A 85 0A 85 EE 85 0A 85 0A 85 EE 01 A1 EE 03 8D    2440  EE 84 02 84 15 84 EE A3 EE A3 EE 04 84 EE 02 84
2250  03 8D EE 04 8B 05 8B EE 43 22 A3 EE A3 EE A3 EE    2450  EE 52 21 A3 EE A3 EE A3 EE 11 85 08 85 EE 11 85
2260  85 19 85 EE 85 19 85 EE 85 19 85 EE 85 19 85 EE    2460  08 85 EE 11 85 08 85 EE 96 08 85 EE A3 EE 11 92
2270  88 13 88 EE 88 13 88 EE 88 13 88 EE 44 1F A3 EE    2470  EE 11 92 EE 53 36 04 81 0F 8B EE 03 82 0E 8D EE
2280  A3 EE A3 EE 85 19 85 EE 85 19 85 EE 85 19 85 EE    2480  01 84 0D 90 EE 85 0B 85 09 85 EE 85 0B 85 09 85
2290  85 19 85 EE 02 9F EE 03 9D EE 04 9B EE 45 2C A3    2490  EE 85 0B 85 09 85 EE 85 0B 85 09 85 EE 01 93 0A
22A0  EE A3 EE A3 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85    24A0  84 EE 03 8F 0C 82 EE 04 8D 0D 81 EE 54 19 EE 1E
22B0  EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE 85 0A 85    24B0  85 EE 1E 85 EE 1E 85 EE A3 EE A3 EE A3 EE 1E 85
22C0  0A 85 EE 85 19 85 EE 85 19 85 EE 46 25 A3 EE A3    24C0  EE 1E 85 EE 1E 85 EE 55 14 A3 EE A3 EE A3 EE 85
22D0  EE A3 EE 0F 85 0A 85 EE 0F 85 0A 85 EE 0F 85 0A    24D0  EE 85 EE 85 EE 85 EE A3 EE A3 EE A3 EE 56 1C 18
22E0  85 EE 0F 85 0A 85 EE 0F 85 0A 85 EE 1E 85 EE 1E    24E0  8B EE 11 8A EE 0B 89 EE 05 89 EE 88 EE 88 EE 05
22F0  85 EE 47 26 A3 EE A3 EE A3 EE 85 19 85 EE 85 19    24F0  89 EE 0B 89 EE 11 8A EE 18 8B EE 57 18 A3 EE A3

2300  85 EE 85 08 84 0D 85 EE 85 08 84 0D 85 EE 91 0A    2500  EE 8A EE 07 8D EE 11 92 EE 11 92 EE 07 8D EE 8A
2310  88 EE 91 0A 88 EE 91 0A 88 EE 48 18 A3 EE A3 EE    2510  EE A3 EE A3 EE 58 2C 88 15 86 EE 05 86 0F 86 EE
2320  A3 EE 0F 85 EE 0F 85 EE 0F 85 EE 0F 85 EE A3 EE    2520  08 86 09 86 EE 0B 86 03 86 EE 0E 89 EE 0E 89 EE
2330  A3 EE A3 EE 49 1F EE 85 19 85 EE 85 19 85 EE 85    2530  0B 86 03 86 EE 08 86 09 86 EE 05 86 0F 86 EE 88
2340  19 85 EE A3 EE A3 EE A3 EE 85 19 85 EE 85 19 85    2540  15 86 EE 59 1C 1A 89 EE 17 88 EE 14 86 EE 12 85
2350  EE 85 19 85 EE 4A 1A 04 87 EE 03 88 EE 02 89 EE    2550  EE 94 EE 94 EE 12 85 EE 14 86 EE 17 88 EE 1A 89
2360  84 EE 84 EE 84 EE 84 EE 02 A1 EE 03 A0 EE 04 9F    2560  EE 5A 34 86 18 85 EE 89 15 85 EE 85 01 86 12 85
2370  EE 4B 2A A3 EE A3 EE 0F 85 EE 0D 84 01 84 EE 0B    2570  EE 85 04 86 0F 85 EE 85 07 86 0C 85 EE 85 0A 86
2380  84 05 84 EE 09 84 09 84 EE 07 84 0D 84 EE 05 84    2580  09 85 EE 85 0D 86 06 85 EE 85 10 86 03 85 EE 85
2390  11 84 EE 03 84 15 84 EE 01 84 19 84 EE 4C 14 A3    2590  13 8B EE 85 16 88 EE 30 1C A3 EE A3 EE A3 EE 85
23A0  EE A3 EE A3 EE 85 EE 85 EE 85 EE 85 EE 85 EE 85    25A0  19 85 EE 85 19 85 EE 85 19 85 EE 85 19 85 EE A3
23B0  EE 85 EE 4D 18 A3 EE A3 EE 17 8C EE 0C 8E EE 8F    25B0  EE A3 EE A3 EE 31 0D EE EE EE A3 EE A3 EE A3 EE
23C0  EE 8F EE 0C 8E EE 17 8C EE A3 EE A3 EE 4E 19 A3    25C0  EE EE EE EE 32 30 94 0A 85 EE 94 0A 85 EE 94 0A
23D0  EE A3 EE 1A 89 EE 13 8A EE 0D 89 EE 0D 89 EE 06    25D0  85 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE 85 0A
23E0  89 EE 89 EE A3 EE A3 EE 4F 20 A3 EE A3 EE 84 1B    25E0  85 0A 85 EE 85 0A 85 0A 85 EE 85 0A 94 EE 85 0A
23F0  84 EE 84 1B 84 EE 84 1B 84 EE 84 1B 84 EE 84 1B    25F0  94 EE 85 0A 94 EE 33 30 85 0A 85 0A 85 EE 85 0A
```

```
2600
~~3400~~ 85 0A 85 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE
3410  85 0A 85 0A 85 EE 85 0A 85 0A 85 EE 85 0A 85 0A
3420  85 EE A3 EE A3 EE A3 EE 34 1B 10 93 EE 10 93 EE
3430  10 93 EE 10 85 EE 10 85 EE 10 85 EE A3 EE A3 EE
3440  A3 EE 10 85 EE 35 30 85 0A 94 EE 85 0A 94 EE 85
3450  0A 94 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE 85
3460  0A 85 0A 85 EE 85 0A 85 0A 85 EE 94 0A 85 EE 94
3470  0A 85 EE 94 0A 85 EE 36 2A A3 EE A3 EE A3 EE 85
3480  08 85 0C 85 EE 85 08 85 0C 85 EE 85 08 85 0C 85
3490  EE 85 08 85 0C 85 EE 92 0C 85 EE 92 0C 85 EE 92
34A0  0C 85 EE 37 1B 1B 88 EE 1B 88 EE 1B 88 EE 1E 85
34B0  EE 1E 85 EE 1E 85 EE 1E 85 EE A3 EE A3 EE A3 EE
34C0  38 24 A3 EE A3 EE A3 EE 85 0A 85 0A 85 EE 85 0A
34D0  85 0A 85 EE 85 0A 85 0A 85 EE 85 0A 85 0A 85 EE
34E0  A3 EE A3 EE A3 EE 39 23 11 92 EE 11 92 EE 11 92
34F0  EE 11 85 08 85 EE 11 85 08 85 EE 11 85 08 85 EE
[illegible]
3500  11 85 08 85 EE A3 EE A3 EE A3 EE 0D 07 EE EE EE
3510  EE EE EE EE 2A 34 EE 09 82 06 82 06 82 EE 0B 82
3520  04 82 04 82 EE 0D 82 02 82 02 82 EE 0F 86 EE 09
3530  92 EE 0F 86 EE 0D 82 02 82 02 82 EE 0B 82 04 82
3540  04 82 EE 09 82 06 82 06 82 EE 2E 0D EE EE EE EE
3550  85 EE 85 EE 85 EE EE EE EE 2D 1E 0F 85 EE 0F 85
3560  EE 0F 85 EE 0F 85 EE 0F 85 EE 0F 85 EE 0F 85 EE
3570  0F 85 EE 0F 85 EE 0F 85 EE 2B 1C EE 0F 85 EE 0F
```

```
27
3580  85 EE 0F 85 EE 05 99 EE 05 99 EE 05 99 EE 0F 85
3590  EE 0F 85 EE 0F 85 EE 3A 16 EE EE EE EE 05 85 0D
35A0  85 EE 05 85 0D 85 EE 05 85 0D 85 EE EE EE EE 21
35B0  13 EE EE EE EE 85 03 9B EE 85 03 9B EE 85 03 9B
35C0  EE EE EE EE 3B 16 EE EE EE EE 05 85 0E 85 EE 04
35D0  86 0E 85 EE 02 88 0E 85 EE EE EE EE 3F 27 17 8C
35E0  EE 17 8C EE 17 8C EE 1E 85 EE 85 02 8B 0C 85 EE
35F0  85 02 8B 0C 85 EE 85 02 8B 0C 85 EE 0D 96 EE 0D
[illegible]
3600  96 EE 0D 96 EE 24 30 02 85 08 92 EE 02 85 08 92
3610  EE 02 85 08 92 EE 02 85 08 85 08 85 EE A3 EE A3
3620  EE 02 85 08 85 08 85 EE 02 92 08 85 EE 02 92 08
3630  85 EE 02 92 08 85 EE 2C 0F EE EE EE EE 03 85 EE
3640  02 86 EE 88 EE EE EE EE 28 16 EE EE EE EE A3 EE
3650  A3 EE A3 EE 85 19 85 EE 85 19 85 EE 85 19 85 EE
3660  29 16 85 19 85 EE 85 19 85 EE 85 19 85 EE A3 EE
3670  A3 EE A3 EE EE EE EE EE FF CD 00 40 5E 40 C0 30
3680  4E 1A 40 40 CF 75 C4 82 4F 6F 46 00 56 76 06 00
3690  96 75 52 40 DF C6 76 A0 6E 77 04 42 5F FF C6 80
36A0  15 3E 40 44 57 F4 07 37 F6 4F 40 04 6B EE E7 02
36B0  F7 E2 40 30 86 37 C3 05 C7 76 62 01 96 9F E6 62
36C0  47 5D 44 01 76 23 04 61 47 D3 02 01 B2 3E 00 00
36D0  97 F6 12 A0 F7 D7 40 00 97 6C 82 40 E7 F6 00 88
36E0  B7 64 44 21 BA 0E 02 01 0F FD 00 01 A9 C1 00 00
~~36F0~~ FF FE 06 C0 76 F5 C6 D0 FF D6 00 40 A7 FC 42 60
28F0
```

EDITORS NOTE: The disassembler program was originally written for the Apple and appeared in Doctor Dobbs Journal (Sept 76). It has been modified for KIM by Bob Kurtz and your editor. Bob Kurtz wrote the article.

KIM-1 "DISASSEMBLER" PROGRAM

Bob Kurtz
Micro-Z Co
P O Box 2426
Rolling Hills Ca 90274

PRELIMINARY:

The purpose of the disassembler is to take any program that has been entered into memory in the KIM-1, and to print-out an "object" code and a "source" code listing of this program - to permit analysis and modification, if desired. In a sense, it takes a completed program and reconstructs the assembly language format - or "dis-assembles" the program.

The following is a sample of the print-out format:

```
Address     Object Code     Source Code

 23BC-      E8              INY
 23BD-      A9 53           LDA    #53
 23BF-      85 01           STA    01
 23C1       91 7E           STA    (7E),Y
 23C3-      4C 64 1C        JMP    1C64
```

The Address and Object Code columns are the standard listings for the program under scrutiny. You will notice that the disassembler has arranged the code listing by one, two, or three byte commands and has printed the address column accordingly.

The Source Code columns contain the MOS Technology 650X Mnemonic abbreviations for the command - and the Operand listing. The following is an explanation of the address mode for the various operands:

Operand	Address Mode
blank	Accumulator, Implied
#53	Immediate
01	Zero Page
01,X 01,Y	Zero Page, indexed by X or Y
(7E),Y	Indirect Indexed
(7E,X)	Indexed Indirect
1C64	Absolute of Branch
(1C64)	Indirect
1C64,X	Absolute (indexed by X)
1C64,Y	Absolute (indexed by Y)

PROCEDURE:

1. Load the starting address of the program to be disassembled into locations 0000 (Low byte) and 0001 (High byte).
2. Go to location 2000
3. Press "G" on terminal

The "disassembler" will now print-out the first 13 commands of the program under scrutiny. At the end of this print-out, simply press "G" again and the next 13 commands will be printed out. Continuing to press "G" whenever the program stops, will step you through the entire program under investigation.

The program stops after each 13 commands. If you wish to modify this, change the byte in location 2010 from $0D (13 decimal) to any number up to $FF (256 decimal).

If portions of the disassembled program do not appear to make sense, these may be "look-up" tables within the program. As an example, the disassembler can be used to "disassemble" the disassembler program! Addresses $2000 to $211A will print out properly since these contain the body of the program commands. However, locations $211B to $21F9 contain the tables for all the mnemonics and symbols and will print-out gibberish.

```
                ;DISASSEMBLER PROGRAM FOR THE 6502
2000            ;WRITTEN BY STEVE WOZNIAK & ALLEN BAUM
2000            ;AND PUBLISHED IN DOCTOR DOBBS JOURNAL
2000            ;SEPT 1976
2000            ;
2000
2000                              *=$0
0000                      PCL     *=*+1
0001                      PCH     *=*+1
0002                      COUNT   *=*+1
0003                      FORMAT  *=*+1
0004                      LENGTH  *=*+1
0005                      LMNEM   *=*+1
0006                      RMNEM   *=*+1
0007                      YSAVE   *=*+1
0008                      ;
0008                      ;KIM I/O TO FOLLOW
0008                      PRTBYT  =$1E3B
0008                      OUTCH   =$1EA0
0008                      CRLF    =$1E2F
0008                      CLEAR   =$1C64
0008                      OUTSP   =$1E9E
0008                      ;
0008                              *=$2000
2000  20 0F 20            START   JSR DSMBL
2003  20 9E 1E                    JSR OUTSP
2006  20 9E 1E                    JSR OUTSP
2009  20 9E 1E                    JSR OUTSP
200C  4C 64 1C                    JMP CLEAR
200F  A9 0D               DSMBL   LDA #13         ;COUNT FOR 13 INSTR. DSMBLY.
2011  85 02                       STA COUNT
```

```
2013  20 21 20  DSMBL2 JSR INSTDS      ;DISASSEMBLE AND DISPLAY INSTR.
2016  20 FC 20         JSR PCADJ
2019  85 00            STA PCL         ;UPDATE PCL,H TO NEXT INSTR.
201B  84 01            STY PCH
201D  C6 02            DEC COUNT       ;DONE FIRST 19 INSTR?
201F  D0 F2            BNE DSMBL2      ;YES, LOOP. ELSE DSMBL 20TH.
2021  20 E2 20  INSTDS JSR PRPC        ;PRINT PCL,H
2024  A1 00            LDA (PCL,X)     ;GET OPCODE
2026  A8               TAY
2027  4A               LSR A           ;EVEN/ODD TEST
2028  90 0B            BCC IEVEN
202A  4A               LSR A           ;TEST BIT 1.
202B  B0 17            BCS ERR         ;XXXXXX11 INSTR. INVALID.
202D  C9 22            CMP #$22
202F  F0 13            BEQ ERR         ;10001001 INSTR. INVALID.
2031  29 07            AND #$7         ;MASK 3 BITS FOR ADDRESS MODE &
2033  09 80            ORA #$80        ;ADD INDEXING OFFSET.
2035  4A        IEVEN  LSR A           ;LSB INTO CARRY FOR
2036  AA               TAX             ;LEFT/RIGHT TEST BELOW.
2037  BD 1B 21         LDA MODE,X      ;INDEX INTO ADDRESS MODE TABLE.
203A  B0 04            BCS RTMODE      ;IF CARRY SET USE LSD FOR
203C  4A               LSR A           ;PRINT FORMAT INDEX
203D  4A               LSR A
203E  4A               LSR A           ;IF CARRY CLEAR USE MSD.
203F  4A               LSR A
2040  29 0F     RTMODE AND #$F         ;MASK FOR 4-BIT INDEX.
2042  D0 04            BNE GETFMT      ;$0 FOR INVALID OP CODES.
2044  A0 80     ERR    LDY #$80        ;SUBSTITUTE $80 FOR INVALID OP,
2046  A9 00            LDA #$0         ;SET PRINT FORMAT INDEX TO 0
2048  AA        GETFMT TAX
2049  BD 5F 21         LDA MODE2,X     ;INDEX INTO PRINT FORMAT TABLE.
204C  85 03            STA FORMAT      ;SAVE FOR ADDRESS FIELD FORMAT.
204E  29 03            AND #$3         ;MASK 2-BIT LENGTH. 0=1-BYTE
2050  85 04            STA LENGTH      ;1=2-BYTE, 2=3-BYTE.
2052  98               TYA             ;OP CODE.
2053  29 8F            AND #$8F        ;MASK IT FOR 1XXX1010 TEST.
2055  AA               TAX             ;SAVE IT.
2056  98               TYA             ;OP CODE TO 'A' AGAIN.
2057  A0 03            LDY #$3
2059  E0 8A            CPX #$8A
205B  F0 0B            BEQ MNNDX3
205D  4A        MNNDX1 LSR A
205E  90 08            BCC MNNDX3      ;FORM INDEX INTO MNEMONIC TABLE.
2060  4A               LSR A
2061  4A        MNNDX2 LSR A           ;1XXX1010 -> 00101XXX
2062  09 20            ORA #$20        ;XXXYYY01 -> 00111XXX
2064  88               DEY             ;XXXYYY10 -> 00110XXX
2065  D0 FA            BNE MNNDX2      ;XXXYY100 -> 00100XXX
2067  C8               INY             ;XXXXX000 -> 000XXXXX
2068  88        MNNDX3 DEY
2069  D0 F2            BNE MNNDX1
206B  48               PHA             ;SAVE MNEMONIC TABLE INDEX.
206C  B1 00     PROP   LDA (PCL),Y
206E  20 13 21         JSR PRBYT
2071  A2 01            LDX #$1
2073  20 F3 20  PROPBL JSR PRBL2
2076  C4 04            CPY LENGTH      ;PRINT INSTR (1 TO 3 BYTES)
2078  C8               INY             ;IN A 12 CHARACTYER FIELD.
2079  90 F1            BCC PROP
207B  A2 03            LDX #$3         ;CHAR COUNT FOR MNEMONIC PRINT.
207D  C0 04            CPY #$4
207F  90 F2            BCC PROPBL
2081  68               PLA             ;RECOVER MNEMONIC INDEX.
2082  A8               TAY
2083  B9 79 21         LDA MNEML,Y
2086  85 05            STA LMNEM       ;FETCH 3-CHAR MNEMONIC
2088  B9 B9 21         LDA MNEMR,Y     ;(PACKED IN TWO BYTES)
208B  85 06            STA RMNEM
208D  A9 00     PRMN1  LDA #$0
208F  A0 05            LDY #$5
2091  06 06     PRMN2  ASL RMNEM
2093  26 05            ROL LMNEM       ;SHIFT 5 BITS OF CHAR INTO 'A'.
2095  2A               ROL A           ;(CLEAR CARRY)
2096  88               DEY
2097  D0 F8            BNE PRMN2
2099  69 3F            ADC #$3F        ;ADD '?' OFFSET.
209B  20 0B 21         JSR OUTC        ;OUTPUT A CHAR OR MNEMONIC
209E  CA               DEX
209F  D0 EC            BNE PRMN1
20A1  20 F1 20         JSR PRBLNK      ;OUTPUT 3 BLANKS.
20A4  A2 06            LDX #$6         ;COUNT FOR 6 PRINT FORMAT BITS.
20A6  E0 03     PRADR1 CPX #$3
20A8  D0 12            BNE PRADR3      ;IF X=3 THEN PRINT ADDRESS VAL.
20AA  A4 04            LDY LENGTH
20AC  F0 0E            BEQ PRADR3      ;NO PRINT IF LENGTH=0.
20AE  A5 03     PRADR2 LDA FORMAT
20B0  C9 E8            CMP #$E8        ;HANDLE REL ADDRESSING MODE
20B2  B1 00            LDA (PCL),Y     ;SPECIAL (PRINT TARGET ADDR)
20B4  B0 1C            BCS RELADR      ;(NOT DISPLACEMENT)
20B6  20 13 21         JSR PRBYT       ;OUTPUT 1- OR 2- BYTE ADDRESS.
20B9  88               DEY             ;MORE SIGNIFICANT BYTE FIRST.
20BA  D0 F2            BNE PRADR2
```

```
20BC  06 03     PRADR3 ASL FORMAT      ;TEST NEXT PRINT FORMAT BIT.
20BE  90 0E            BCC PRADR4      ;IF 0, DONT PRINT
20C0  BD 6C 21         LDA CHAR1-1,X   ;CORRESPONDING CHAR.
20C3  20 0B 21         JSR OUTC        ;OUTPUT 1 OR 2 CHARS.
20C6  BD 72 21         LDA CHAR2-1,X   ;(IF CHAR FROM CHAR2 IS 0,
20C9  F0 03            BEQ PRADR4      ;DON'T PRINT IT)
20CB  20 0B 21         JSR OUTC
20CE  CA        PRADR4 DEX
20CF  D0 D5            BNE PRADR1
20D1  60               RTS             ;RETURN IF DONE 6 FORMAT BITS.
20D2  20 FF 20  RELADR JSR PCADJ3      ;PCL,H + DISPL + 1 TO 'A','Y'.
20D5  AA               TAX
20D6  E8               INX
20D7  D0 01            BNE PRNTYX      ;  +1 TO 'X','Y'.
20D9  C8               INY
20DA  98        PRNTYX TYA
20DB  20 13 21  PRNTAX JSR PRBYT       ;PRINT TARGET ADDRESS OF BRANCH
20DE  8A        PRNTX  TXA             ;AND RETURN
20DF  4C 13 21         JMP PRBYT
20E2  20 2F 1E  PRPC   JSR CRLF        ;OUTPUT CARRIAGE RETURN.
20E5  A5 01            LDA PCH
20E7  A6 00            LDX PCL
20E9  20 DB 20         JSR PRNTAX      ;OUTPUT PCL,H
20EC  A9 2D            LDA #'-
20EE  20 0B 21         JSR OUTC        ;OUTPUT '-'
20F1  A2 03     PRBLNK LDX #$3         ;BLANK COUNT
20F3  A9 20     PRBL2  LDA #'
20F5  20 0B 21  PRBL3  JSR OUTC        ;OUTPUT A BLANK
20F8  CA               DEX
20F9  D0 F8            BNE PRBL2       ;LOOP UNTIL COUNT =0
20FB  60               RTS
20FC  A5 04     PCADJ  LDA LENGTH      ;0=1-BYTE, 1=2-BYTE, 2=3-BYTE.
20FE  38        PCADJ2 SEC
20FF  A4 01     PCADJ3 LDY PCH
2101  AA               TAX             ;TEST DISPL SIGN (FOR REL
2102  10 01            BPL PCADJ4      ;BRANCH). EXTEND NEG

2104  88               DEY             ;BY DECREMENTING PCH.
2105  65 00     PCADJ4 ADC PCL
2107  90 01            BCC RTS1        ;PCL+LENGTH (OR DISPL) +1 TO 'A'.
2109  C8               INY             ;CARRY INTO 'Y' (PCH)
210A  60        RTS1   RTS
210B  84 07     OUTC   STY YSAVE
210D  20 A0 1E         JSR OUTCH
2110  A4 07            LDY YSAVE
2112  60               RTS
2113  84 07     PRBYT  STY YSAVE
2115  20 3B 1E         JSR PRTBYT
2118  A4 07            LDY YSAVE
211A  60               RTS
211B                   ;
211B                   ;
211B                   ;THE TABLES FOLLOW---
211B                   ;
211B  40        MODE   .BYTE $40,$2,$45,$3,$D0,$8,$40,$9,$30
211C  02
211D  45
211E  03
211F  D0
2120  08
2121  40
2122  09
2123  30
2124                   ;XXXXXXZ0 INSTRUCTIONS.
2124  22               .BYTE $22,$45,$33,$D0,$8,$40,$9,$40,$2,$45
2125  45
2126  33
2127  D0
2128  08
2129  40
212A  09
212B  40
212C  02
212D  45
212E                   ;Z=0, LEFT HALF BYTE
212E                   ;Z=1, RIGHT HALF BYTE
212E  33               .BYTE $33,$D0,$8,$40,$9,$40,$2,$45,$B3,$D0
212F  D0
2130  08
2131  40
2132  09
2133  40
2134  02
2135  45
2136  B3
2137  D0
2138  08               .BYTE $8,$40,$9,$0,$22,$44,$33,$D0,$8C,$44
2139  40
213A  09
213B  00
213C  22
213D  44
213E  33
213F  D0
2140  8C
2141  44
```

```
2142  00          .BYTE $0,$11,$22,$44,$33,$D0,$8C,$44,$9A,$10
2143  11
2144  22
2145  44
2146  33
2147  D0
2148  8C
2149  44
214A  9A
214B  10
214C  22          .BYTE $22,$44,$33,$D0,$8,$40,$9,$10,$22,$44
214D  44
214E  33
214F  D0
2150  08
2151  40
2152  09
2153  10
2154  22
2155  44
2156  33          .BYTE $33,$D0,$8,$40,$9,$62
2157  D0
2158  08
2159  40
215A  09
215B  62
215C        ;YYXXXZ01 INSTRUCTIONS
215C  13          .BYTE $13,$78,$A9
215D  78
215E  A9
215F  00    MODE2 .BYTE $0         ;ERR
2160  21          .BYTE $21        ;IMM
2161  01          .BYTE $01        ;Z-PAG
2162  02          .BYTE $02        ;ABS
2163  00          .BYTE $0         ;IMPL
2164  80          .BYTE $80        ;ACC
2165  59          .BYTE $59        ;(Z-PAG,X)
2166  4D          .BYTE $4D        ;(Z-PAG),Y
2167  11          .BYTE $11        ;Z-PAG,X
2168  12          .BYTE $12        ;ABS,X
2169  06          .BYTE $6         ;ABS,Y
216A  4A          .BYTE $4A        ;(ABS)
216B  05          .BYTE $5         ;Z-PAG,Y
216C  1D          .BYTE $1D        ;REL
216D  2C    CHAR1 .BYTE $2C,$29,$2C,$23,$28,$41
216E  29
216F  2C
2170  23
2171  28
2172  41
2173  59    CHAR2 .BYTE $59,$0,$58,$00,$0,$0
2174  00
2175  58
2176  00
2177  00
2178  00
2179        ;XXXXX000 INSTRUCTIONS
2179  1C    MNEML .BYTE $1C,$8A,$1C,$23,$5D,$8B,$1B,$A1,$9D
217A  8A
217B  1C
217C  23
217D  5D
217E  8B
217F  1B
2180  A1
2181  9D
2182  8A          .BYTE $8A,$1D,$23,$9D,$8B,$1D,$A1,$0,$29,$19
2183  1D
2184  23
2185  9D
2186  8B
2187  1D
2188  A1
2189  00
218A  29
218B  19
218C  AE          .BYTE $AE,$69,$A8,$19,$23,$24,$53,$1B,$23
218D  69
218E  A8
218F  19
2190  23
2191  24
2192  53
2193  1B
2194  23
2195  24          .BYTE $24,$53,$19,$A1,$0
2196  53
2197  19
2198  A1
2199  00
```

```
1950  219A           ;XXXYY100 INSTRUCTIONS.
1960  219A   1A             .BYTE $1A,$5B,$5B,$A5,$69,$24,$24
1960  219B   5B
1960  219C   5B
1960  219D   A5
1960  219E   69
1960  219F   24
1960  21A0   24
1962  21A1           ;1XXX1010 INSTRUCTIONS
1963  21A1   AE             .BYTE $AE,$AE,$A8,$AD,$29,$0,$7C,$0
1963  21A2   AE
1963  21A3   A8
1963  21A4   AD
1963  21A5   29
1963  21A6   00
1963  21A7   7C
1963  21A8   00
1964  21A9           ;XXXYYY10 INSTRUCTIONS
1965  21A9   15             .BYTE $15,$9C,$6D,$9C,$A5,$69,$29,$53
1965  21AA   9C
1965  21AB   6D
1965  21AC   9C
1965  21AD   A5
1965  21AE   69
1965  21AF   29
1965  21B0   53
1970  21B1           ;XXXYY01 INSTRUCTIONS.
1980  21B1   84             .BYTE $84,$13,$34,$11,$A5,$69,$23,$A0
1980  21B2   13
1980  21B3   34
1980  21B4   11
1980  21B5   A5
1980  21B6   69
1980  21B7   23
1980  21B8   A0
1990  21B9           ;XXXXX000 INSTRUCTIONS.
2000  21B9   D8      MNEMR  .BYTE $D8,$62,$5A,$48,$26,$62,$94,$88
2000  21BA   62
2000  21BB   5A
2000  21BC   48
2000  21BD   26
2000  21BE   62
2000  21BF   94
2000  21C0   88
2010  21C1   54             .BYTE $54,$44,$C8,$54,$68,$44,$E8,$94,$0,$B4
2010  21C2   44
2010  21C3   C8
2010  21C4   54
2010  21C5   68
2010  21C6   44
2010  21C7   E8
2010  21C8   94
2010  21C9   00
2010  21CA   B4
2020  21CB   08             .BYTE $8,$84,$74,$B4,$28,$6E,$74,$F4,$CC,$4A
2020  21CC   84
2020  21CD   74
2020  21CE   B4
2020  21CF   28
2020  21D0   6E
2020  21D1   74
2020  21D2   F4
2020  21D3   CC
2020  21D4   4A
2030  21D5   72             .BYTE $72,$F2,$A4,$8A
2030  21D6   F2
2030  21D7   A4
2030  21D8   8A
2040  21D9           ;XXXYY100 INSTRUCTIONS.
2050  21D9   00             .BYTE $0,$AA,$A2,$A2,$74,$74,$74,$72
2050  21DA   AA
2050  21DB   A2
2050  21DC   A2
2050  21DD   74
2050  21DE   74
2050  21DF   74
2050  21E0   72
2060  21E1           ;1XXX1010 INSTRUCTIONS.
2070  21E1   44             .BYTE $44,$68,$B2,$32,$B2,$0,$22,$0
2070  21E2   68
2070  21E3   B2
2070  21E4   32
2070  21E5   B2
2070  21E6   00
2070  21E7   22
2070  21E8   00
2080  21E9           ;XXXYYY10 INSTRUCTIONS.
2090  21E9   1A             .BYTE $1A,$1A,$26,$26,$72,$72,$88,$C8
2090  21EA   1A
2090  21EB   26
2090  21EC   26
2090  21ED   72
2090  21EE   72
2090  21EF   88
2090  21F0   C8
```

```
21F1              ;XXXYYY01 INSTRUCTIONS
21F1  C4                    .BYTE $C4,$CA,$26,$48,$44,$44,$A2,$C8
21F2  CA
21F3  26
21F4  48
21F5  44
21F6  44
21F7  A2
21F8  C8
21F9              FINISH .END
```

CHECK-OUT

by Robert D. Larrabee
18801 Woodway Drive
Derwood, MD 20855

Did you ever want to check-out a new program without having to continually hit the plus key of your KIM? Did you ever wish you could back-up a byte or two? Did you ever want to add some material in the middle of a program without having to reenter all of the succeeding bytes? If you ever did, program CHECK-OUT is for you!

Load this program into the stack page, push the PC key to enter the starting address (0100), and then push the GO key. The display will show address 0000 and the contents of that memory location. Then push the B key to start automatically scanning through memory twoard the Back of the program, or the F key to start scanning toward the Front of the program. The keys 1 through 9 control the scan speed. The zero key stops the scan at the displayed address. The A key stops the scan one position beyond the currently displayed address, while the E key stops the scan one position previous to the currently displayed address. If the scan speed is set to 9 (the fastest possible), the A and E keys are equivalent to an immediate one step forward or backward. Because the plus key was there, it was given its normal function.

If, in scanning a program, an error is found, stop the scan at the error (by pushing the zero key at the error, or stepping there with the A, E, or plus keys). The push the DA key, and you will enter the KIM monitor in data mode. The corrected data can then be immediately entered from the keyboard, and you can then return to program CHECK-OUT by pushing the ST key.

If you wish to make a large jump in address, and scanning there at speed 9 would take too long, push the AD key to enter the KIM monitor in address mode. The new address of interest can then be immediately entered from the keyboard, and you can then return to program CHECK-OUT by pushing the ST key.

The C key Creates spaces for additional bytes in a program by moving the program material down from the displayed address one unit each time the C key is depressed. The byte displayed and all following bytes are moved down and the created space at the displayed address is filled with zeros. The D key Deletes the displayed byte by moving up all the following program material one unit each time the D key is depressed. Neither the C or D key effects the portion of the program before the displayed address. The table below shows what address program CHECK-OUT considers to be the end of the program. Notice that this depends on the page currently being displayed when the C or D key is depressed. If desired, these ending locations can be changed by entering appropriate new address information at the locations indicated in the last two columns of the table.

When attempting to do something program CHECK-OUT considers illegal (for example, modifying some of its own instructions on page 1), the display will go blank for as long as the illegal key is depressed, and then all will return to the previous conditions when the illegal key is released.

Program CHECK-OUT does nothing in the way of changing branch instruction addresses when creating or deleting spaces. If this feature is desired, I invite you to write a routine to perform this address manipulation. Change the branch instruction at location 01D1 in program CHECK-OUT to branch to your routine, and then return to address 0104 in program CHECK-OUT at the end of your routine.

Page of current address	Address taken as the end of the program	Location of these data within program CHECK-OUT	
		High	Low
0	00EE	-	0187
1	not allowed	-	-
2-16	03FF	0193	0195
17	17E6	-	0199
above 17	not allowed	-	-

STORAGE AREA

```
00EF 00            PCL          program counter, low
00F0 01            PCH          program counter, high
00F1 00            PREG         status register
00F2 FF            SPUSER       stack pointer
00F3 00            ADL          current address, low
00F4 00            ADH          current address, high
00F5 05            SS           scan speed (0-9)
00F6 00            RD           read disable (disable if not zero)
00F7 00            MODE         mode of operation (stop scan if zero)
00F8 FF            LC           loop counter for display loop
00F9 00            INH          data position for display
00FA FA            POINTL       pointer address, low
00FB 00            POINTH       pointer address, high
00FC 01            TEMP1        temporary storage register #1
00FD 00            TEMP2        temporary storage register #2
00FE 00                         not used
00FF 00                         not used
```

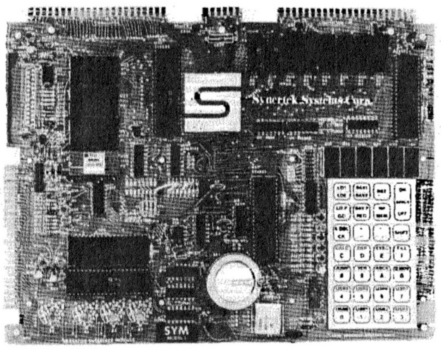
this program assumes the CPU is in the binary mode
$00F1 = $00

MAIN PROGRAM STARTS HERE

```
0100 A9 00     ENTRY  LDA #00        initial entry into program
0102 85 FB            STA POINTH     set initial value of POINTL
0104 A5 FA     START  LDA POINTL     get pointer address, low
0106 85 F3            STA ADL        store pointer address, low in ADL
0108 A5 FB            LDA POINTH     get pointer address, high
010A 85 F4            STA ADH        store pointer address, high in ADH
010C A2 FF            LDX #FF        initial value of SPUSER and LC
010E 9A               TXS            initialize stack pointer to #FF
010F 86 F8            STX LC         initialize loop counter to #FF
0111 A2 04            LDX #04        value for interrupt vector, low
0113 8E FA 17         STX 17FA       initialize interrupt vector, low
0116 A2 01            LDX #01        value for interrupt vector, high
0118 8E FB 17         STX 17FB       initialize interrupt vector, high
```

SYM-1, 6502-BASED MICROCOMPUTER

- FULLY-ASSEMBLED AND COMPLETELY INTEGRATED SYSTEM that's ready-to-use
- ALL LSI IC'S ARE IN SOCKETS
- 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, expandable to five on-board.
- 4K BYTE ROM RESIDENT MONITOR and Operating Programs.
- Single 5 Volt power supply is all that is required.
- 1K BYTES OF 2114 STATIC RAM onboard with sockets provided for immediate expansion to 4K bytes onboard, 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 1500 Baud)
  —Full duplex 20mA Teletype Interface
  —System Expansion Bus Interface
  —TV Controller Board Interface
  —CRT Compatible Interface (RS-232)
- APPLICATION PORT: 15 Bi-directional TTL Lines for user applications with expansion capability for added lines
- EXPANSION PORT FOR ADD-ON MODULES (51 I/O Lines included in the basic system)
- SEPARATE POWER SUPPLY connector for easy disconnect of the d-c power
- AUDIBLE RESPONSE KEYPAD



Synertek has enhanced KIM-1* software as well as the hardware. The software has simplified the user interface. The basic SYM-1 system is programmed in machine language. Monitor status is easily accessible, and the monitor gives the keypad user the same full functional capability of the TTY user. The SYM-1 has everything the KIM-1* has to offer, plus so much more that we cannot begin to tell you here. So, if you want to know more, the SYM-1 User Manual is available, separately.

SYM-1 Complete w/manuals	**$269.00**
SYM-1 User Manual Only	**7.00**
SYM-1 Expansion Kit	**75.00**

Expansion includes 3K of 2114 RAM chips and 1-6522 I/O chip.

SYM-1 Manuals: The well organized documentation package is complete and easy-to-understand.

SYM-1 CAN GROW AS YOU GROW. Its the system to BUILD-ON. Expansion features that are soon to be offered:

***BAS-1 8K Basic ROM (Microsoft)**	**$159.00**
***KTM-2 TV Interface Board**	**349.00**

**We do honor Synertek discount coupons*

QUALITY EXPANSION BOARDS DESIGNED SPECIFICALLY FOR KIM-1, SYM-1 & AIM 65

These boards are set up for use with a regulated power supply such as the one below, but, provisions have been made so that you can add onboard regulators for use with an unregulated power supply. But, because of unreliability, we do not recommend the use of onboard regulators. All I.C.'s are socketed for ease of maintenance. *All boards carry full 90-day warranty.*

All products that we manufacture are designed to meet or exceed industrial standards. All components are first qualtiy and meet full manufacturer's specifications. All this and an extended burn-in is done to reduce the normal percentage of field failures by up to 75%. To you, this means the chance of inconvenience and lost time due to a failure is very rare; but, if it should happen, we guarantee a turn-around time of less than forty-eight hours for repair.

Our money back guarantee: If, for any reason you wish to return any board that you have purchased directly from us within ten (10) days after receipt, complete, in original condition, and in original shipping carton; we will give you a complete credit or refund less a $10.00 restocking charge per board.

VAK-1 8-SLOT MOTHERBOARD

This motherboard uses the KIM-4* bus structure. It provides eight (8) expansion board sockets with rigid card cage. Separate jacks for audio cassette, TTY and power supply are provided. Fully buffered bus.

VAK-1 Motherboard **$129.00**

VAK-2/4 16K STATIC RAM BOARD

This board using 2114 RAMs is configured in two (2) separately addressable 8K blocks with individual write-protect switches.

VAK-2 16K RAM Board with only 8K of RAM (½ populated)	**$239.00**
VAK-3 Complete set of chips to expand above board to 16K	**$175.00**
VAK-4 Fully populated 16K RAM	**$379.00**

VAK-5 2708 EPROM PROGRAMMER

This board requires a +5 VDC and ±12 VDC, but has a DC to DC multiplyer so there is no need for an additional power supply. All software is resident in on-board ROM, and has a zero-insertion socket.

VAK-5 2708 EPROM Programmer **$269.00**

VAK-6 EPROM BOARD

This board will hold 8K of 2708 or 2758, or 16K of 2716 or 2516 EPROMs. *EPROMs not included.*

VAK-6 EPROM Board **$129.00**

VAK-7 COMPLETE FLOPPY-DISK SYSTEM (May '79)

VAK-8 PROTYPING BOARD

This board allows you to create your own interfaces to plug into the motherboard. Etched circuitry is provided for regulators, address and data bus drivers; with a large area for either wire-wrapped or soldered IC circuitry.

VAK-8 Protyping Board **$49.00**

POWER SUPPLIES

ALL POWER SUPPLIES are totally enclosed with grounded enclosures for safety, AC power cord, and carry a full 2-year warranty.

FULL SYSTEM POWER SUPPLY

This power supply will handle a microcomputer and up to 65K of our VAK-4 RAM. ADDITIONAL FEATURES ARE: Over voltage Protection on 5 volts, fused, AC on/off switch. Equivalent to units selling for $225.00 or more

Provides +5 VDC @ 10 Amps & ±12 VDC @ 1 Amp
VAK-EPS Power Supply **$125.00**

KIM-1* Custom P.S. provides 5 VDC @ 1.2 Amps and +12 VDC @ .1 Amps
KCP-1 Power Supply **$41.50**

SYM-1 Custom P.S. provides 5 VDC @ 1.4 Amps
VCP-1 Power Supply **$41.50**

*KIM is a product of MOS Technology

RNB ENTERPRISES INCORPORATED
2967 W. Fairmount Avenue
Phoenix AZ 85017
(602)265-7564

master charge VISA

DISPLAY/KEYBOARD-DECODE LOOP

```
011B A9 80     LOOP1   LDA #80       value to store in the timer
011D 8D 47 17          STA 1747      start timer
0120 20 19 1F  AGAIN   JSR SCAND     activate display
0123 D0 04             BNE ONE       branch if any key is depressed
0125 85 F6             STA RD        enable the reading of the keyboard
0127 F0 43             BEQ DONE      unconditional branch to end of loop
0129 A5 F6     ONE     LDA RD        get value of read disable
012B D0 3F             BNE DONE      branch to end of loop if RD not zero
012D E6 F6             INC RD        set read disable to #01
012F 20 6A 1F  ERROR   JSR GETKEY    read keyboard
0132 C9 00             CMP #00       is the zero key depressed?
0134 F0 32             BEQ MZERO     branch if the zero key is depressed
0136 AA                TAX           store keycode in X register
0137 38                SEC           required by next instruction
0138 E9 0A             SBC #0A       subtract #0A from keycode
013A 10 06             BPL TWO       branch if keycode is not speed control
013C 49 FF             EOR #FF       compute new value of speed
013E 85 F5             STA SS        store new value of speed in SS
0140 10 2A             BPL DONE      unconditional branch to end of loop

0142 8A        TWO     TXA           return keycode to accumulator
0143 C9 12             CMP #12       is the plus key depressed?
0145 F0 36             BEQ PLUS      branch if plus key depressed
0147 10 23             BPL DONE      ignore unused keys
0149 C9 10             CMP #10       is the AD key depressed?
014B D0 03             BNE THREE     branch if AD key not depressed
014D 4C 7C 1C  KIM     JMP 1C7C      enter the KIM monitor
0150 29 0A     THREE   AND #0A       result not #0A for C, D, & DA keys
0152 C9 0A             CMP #0A       result is #0A for A, B, E, & F keys
0154 F0 14             BEQ MODE      branch if key is A, B, E, or F
0156 A4 FB             LDY POINTH    load page number into Y register
0158 88                DEY           decrement page number in Y register
0159 F0 D4             BEQ ERROR     blank display if page number = 1
015B A5 F7             LDA MODE      load mode into Y register
015D F0 02             BEQ FOUR      branch if mode is zero
015F 10 CE             BPL ERROR     blank display if mode not zero
0161 8A        FOUR    TXA           return keycode to accumulator
0162 C9 11             CMP #11       is the DA key depressed?
0164 F0 E7             BEQ KIM       enter the KIM monitor if DA key
0166 D0 1C             BNE MOVE      branch if C or D key depressed
0168 A2 00     MZERO   LDX #00       the new value of mode
016A 86 F7     MODE    STX MODE      store the new value of mode
016C AD 47 17  DONE    LDA 1747      test the clock
016F F0 AF             BEQ AGAIN     display again if not time
0171 E6 F8             INC LC        add one to loop counter
0173 A5 F8             LDA LC        get new value of loop counter
0175 C5 F5             CMP SS        compare to total number desired
0177 90 A2             BCC LOOP1     do another loop if necessary
0179 A6 F7             LDX MODE      get value of mode
017B F0 04             BEQ FIVE      return to START if mode is zero
017D 8A                TXA           put mode in accumulator
017E 20 D6 01          JSR NEXT      set up next POINTL and POINTH
0181 4C 04 01  FIVE    JMP START     end of display/keyboard loop
```

ROUTINE TO SERVICE THE C AND D KEYS

```
0184 86 FC     MOVE    STX TEMP1     store keycode in TEMP1
0186 A0 EE             LDY #EE       location of useful end of page zero
0188 A5 F4             LDA ADH       put page number in accumulator
018A F0 0E             BEQ SEVEN     branch if page zero
018C C9 17             CMP #17       is page number equal to 17?
018E F0 08             BEQ SIX       branch if page number is 17
0190 B0 9D             BCS ERROR     blank display if page number > 17
0192 A9 03             LDA #03       end page in unexpanded KIM system RAM
0194 A0 FF             LDY #FF       location of end of page
0196 30 02             BMI SEVEN     unconditional jump
0198 A0 E6     SIX     LDY #E6       location of useful end of page 17
019A 84 FA     SEVEN   STY POINTL    store end of page location in POINTL
019C 85 FB             STA POINTH    store end page number in POINTH
019E A0 00             LDY #00       value needed for (indirect),Y addressing
01A0 B1 FA             LDA POINTL,Y  LDA with byte at the end of the last page
01A2 AA                TAX           store byte in the X register
01A3 98                TYA           bring #00 to accumulator
01A4 91 FA             STA POINTL,Y  put zero in end of page location
01A6 20 EE 01  LOOP2   JSR SUB       decrement POINTL/POINTH
01A9 A5 FC             LDA TEMP1     get keycode from TEMP1
01AB C9 0D             CMP #0D       which key (C or D) is depressed?
01AD F0 0B             BEQ DKEY      branch if D key is depressed
01AF B1 FA     CKEY    LDA POINTL,Y  get byte at pointed location
01B1 C8                INY           increment Y register to unity
01B2 91 FA             STA POINTL,Y  store byte in next larger address
01B4 88                DEY           decrement Y register to zero
01B5 98                TYA           set accumulator to zero
01B6 91 FA             STA POINTL,Y  store zero in pointed location
01B8 F0 09             BEQ TEST      branch around D key instructions
01BA B1 FA     DKEY    LDA POINTL,Y  get byte at pointed location
01BC 85 FD             STA TEMP2     store temporarily in TEMP2
01BE 8A                TXA           bring following byte to accumulator
01BF 91 FA             STA POINTL,Y  store byte in pointed location
01C1 A6 FD             LDX TEMP2     retrieve stored byte
01C3 A5 F4     TEST    LDA ADH       ADH is endpoint of POINTH
01C5 C5 FB             CMP POINTH    has final page been reached?
01C7 90 DD             BCC LOOP2     if not, loop back
01C9 D0 06             BNE END       all done if ADH > POINTH
```

```
01CB A5 F3             LDA ADL        ADL is endpoint of POINTL
01CD C5 FA             CMP POINTL     has final location been reached?
01CF 90 D5             BCC LOOP2      if not, loop back
01D1 4C 04 01  END     JMP START      end of C and D key routine
01D4 EA                NOP            not used
01D5 EA                NOP            not used

                  SUBROUTINE TO STEP POINTL AND POINTH

01D6 A0 00     NEXT    LDY #00        possible new value of mode
01D8 29 01             AND #01        is least significant digit zero?
01DA D0 02             BNE EIGHT      F and B keys require nonzero mode
01DC 84 F7             STY MODE       A and E keys require zero mode
01DE 88        EIGHT   DEY            decrement Y register to #FF
01DF 8A                TXA            bring mode to accumulator
01E0 29 04             AND #04        in which direction is the scan?
01E2 D0 0A             BNE SUB        branch if E or F key depressed
01E4 98        ADD     TYA            bring #FF to accumulator
01E5 C5 FA             CMP POINTL     is POINTL equal to #FF?
01E7 D0 02             BNE NINE       branch if not equal to #FF
01E9 E6 FB             INC POINTH     increment POINTH to new value
01EB E6 FA     NINE    INC POINTL     increment POINTL to new value
01ED 60                RTS            return
01EE C6 FA     SUB     DEC POINTL     decrement POINTL to new value
01F0 A9 FF             LDA #FF        needed for next instruction
01F2 C5 FA             CMP POINTL     is POINTL equal to #FF?
01F4 D0 02             BNE RETURN     branch if not equal to #FF
01F6 C6 FB             DEC POINTH     decrement POINTH to new value
01F8 60        RETURN  RTS            return

                          RESERVED FOR STACK

01F9 to 01FF
```

LANGUAGE LAB

basic

BASIC MOD & PROGRAMMING HINT

by Heinz Joachim Schilling, DJ1XK
Im Grun 15
D-7750 Konstanz 16
W Germany

Two days ago my copy of 6502 User Notes arrived, and because of new format and content I *must say: Congratulations !!! You have arranged it* into several sections, so it is quite easy to find an object of special personal interest. In my case it is BASIC.

You rised a BASIC question, and I have the solution. The problem of re-loading programs comes from a programming error of the Microsoft people. They did not realize that

1. the ID = $00 or $FF are only operable if the tape was not saved with an ID of $00 or $FF.
2. loading to a changed address using $FF in $17F9 is not possible with their tapes as they use $FF in $17F9 during save!

You have to change the LDA $FF into LDA $FE at $274 , the $FE is at $2744, and that's all! You should have a look into the listing of the KIM-loader to see that ID=$FF only comes into operation after the compare between the $17F9-ID and the tape-ID, in our case the compare matches and the tape is loaded to the same location as it is saved.

You are right that the BREAK-Test should include the 30 as an opcode.

The KIM BASIC hint regarding inputting only a 'return' is good, and I have another one!

In every case you try to use the cursor control codes in PRINT-statements with a semi-colon at the end you run into trouble: after 72 chars (or any other number you inputted at the cold start) the BASIC thinks the line would be complete you inserts a CR-LF, and off you are. This is true in all sorts of games (3 cushion billiard, life and so on). But the solution is so easy: add a simple POKE, and everything will run:

```
   1000 PRINT CHR$(9);:POKE 22,1:REM CURSOR RIGHT
```

The POKE goes to the memory cell which holds the *position in the line. The storage of the 1 let* the BASIC think it was at the beginning of the line and so the inclusion of the unwanted CR-LF is dropped.

Besides, I have made a little program with subroutines of HYPERTAPE and LOAD in the same form as they are used in the Micro Ade Assembler, that means with optical control of the loading. You can see on the 7-segment-display three different states: SYNC, Loading and seeking (that means if your load was with a fault and the loader looks for another ID). This little program has routines for calling these subroutines, and it can start BASIC and can start other programs at a specific address. I use the loader- and save-subs from BASIC too, and so my BASIC is in the memory in 3 minutes instead of 18! Besides, I have made a disassembler printout of BASIC to allow easier corrections to BASIC. I think it would be possible to PROM the BASIC with trig functions, but this would be only possible in 9K, and you would have to change something in the coldstart routine. You must only make a little correction to the "Want SIN..." routine and take the first address after the PROM as beginning of free memory to $78, $79, e.g. the Want Sin prompting will be deleted.

The renumbering program on page 12 seems to be good, but I want to make one in assembler which could be loaded to my spare memory at $0400 and must not be deleted at the end of the operation.

I just remember another little correction to my BASIC: I changed the location $243A to $E5 which allows the use of the DEL key at my terminal instead of the underline to delete a wrong character.

The most important thing during work with BASIC for nay corrections: a disassembler printout!!!!

Then it is very easy to follow the flow through the interpreter and make little changes.

BASIC OUTPUT PAGING MOD

by Dick Grabowsky
HDE, Inc.
Box 120
Allamuchy NJ 07820

Marvin Dejong asks where to look to make Basic list 16 lines, rather than scoot the program past his eyes so quickly he can't see it.

Microsoft Basic does not use the KIM CRLF routine. Rather, it outputs a CR (OD) and line feed (OA) followed by a number of nulls as defined in LOC $15. These characters are all transfered to the KIM OUTCH routine. In 9-digit Microsoft Basic, the call to KIM OUTCH is located at $2A51. To limit output, we can intercept this call, count the number of times the CR or LF character pass through our intercept, then halt further output until the operator inputs some character from the keyboard.

The following routine does this by counting the line-feed characters and stopping output until a line-feed is entered via the keyboard. As a bonus, let's add an ability to stop output (i.e. terminate the process) when we've found what we want, or have seen enough. The routine maybe placed anywhere in memory that will not be overwritten by Basic or Basic programs/data.

```
LINCNT  CMP #$0A      ;is it a line feed?
        BNE LINC1     ;no, then output
        INC COUNTR    ;yes, incr the counter
        LDA COUNTR    ;and check to see if
        CMP LNCNT     ;it ='s preset line count
        BEQ LINC2     ;yes, -halt output
        LDA #$0A      ;else, reload line feed
LINC1   JMP $1EA0     ;(KIM OUTCH)

LINC2   LDA #0
        STA COUNTR    ;reset the counter
LINC3   JSR $1E5A     ;(KIM GETCH) get a char
        CMP #$0A      ;line feed?
        BEQ LINC1     ;yes, continue listing
        CMP #$0D      ;return?
        BNE LINC3     ;no, ignore
        JMP $0        ;else, jump to "warm start"


COUNTR  *=*+1         ;line feed counter
LNCNT   *=*+1         ;line count
```

To use the routine, set the address of "LINCNT" at $2A52, $2A53. Preset the values of "COUNTR" to zero and "LNCNT" to the desired number of lines (use HEX). All output will then be limited to the number of lines defined by LNCNT. Since Basic does not use zero page locations from $DD up, $DD and $DE may be good locations to put "COUNTR" and "LNCNT" since this can be done before loading Basic from tape.

When in Basic, the routine can be used to limit output to a specific number of lines by "poke" ing the values of "COUNTR" and "LNCNT" as appropriate.

One additional note, 9 digit Basic sets the base address for programs at locations $4148 and $414A to 41 and 40, respectively. To gain room above Basic for additional software, changing these locations to the address values desired will do the trick. By the way, to check on the authors of Microsoft Basic, enter an "A" in response to the "memory size?" question.

RENUMBER ADDENDUM AND SOME MODS

by Harvey Herman
Dept. of Chemistry
Univ. of North Carolina
Greensboro, NC 27412

I recently sent you a BASIC renumbering program for KIM Microsoft 8K BASIC. In the accompanying letter I noted one restriction about the number of digits in the new line number. Sean McKenna has written me about one further restriction. He notes that numbers after THEN in an assignment, for example, will always be renumbered. Thus, the 2 in 10 IF A = 1 THEN X=2 could be inadvertently changed. He suggests using variables in assignments after THEN (X = V, where V = 2) to avoid the problem.

I also mentioned another renumbering program I use which utilizes a paper tape punch. This program does not have the above problem. If readers are interested in this program I will send it to them on receipt of a SASE and extra loose stamp.

I wrote you recently about the question you posed to your readers about Microsoft BASIC. It is possible to automate the "Y" answer in an initialization routine and I documented the procedure. A further point-4146 can be changed to 4E or 41 if "N" or "A" is the desired response.

In reply to a question posed to me in a private letter I have figured out how to skip all 3 initialization questions and their accompaning messages.

1. To size memory automaticallv change-

Locations	From	To	
40B9	20	4C	JMP $40CD
40BA	18	CD	
40BB	2A	40	

2. To keep 72 as terminal width

410A	20	4C	JMP $4136
410B	18	36	
410C	2A	41	

3. To answer "Y" to trig functions question <u>and</u> skip message

413A	20	4C	JMP $4145
413B	18	45	
413C	2A	41	

4145	C0	A9	LDA #'Y or 'N or 'A
4146	00	59 or 4E or 41	

AUTOMATIC LINE NUMBER ENTRY PROMPT FOR BASIC

```

                 Sean McKenna
                 64 Fairview Ave.
                 Piedmont, CA 94610
                 January, 1979


                 NUMBER REVISED VERSION

                 An automatic line numbering input routine
                 for 9 digit KIM BASIC.  From command BASIC
                 enter # nnnn+ii(sp)CR to begin automatic
                 line number sequencing with nnnn.  Each
                 line will be incremented by ii.  To return
                 to command BASIC enter CR after line number.
                 On delete the line number will be repeated
                 on the next line of the terminal.  Don't
                 forget the space after the increment number
                 or you may get no increment at all or a
                 strange one.  Because of the decimal add
                 the highest line number possible with the
                 program is 9999 and 99 is the highest
                 increment

                  Initialization: SEQFLG and TIMNUM must be
                  initialized to $00 before using the routine

```

```
0200            NUMBER ORG    $0200
0200            INCR   *      $00E5
0200            SEQFLG *      $00E6
0200            TIMNUM *      $00E7
0200            HI     *      $00E8
0200            LO     *      $00E9
0200            BASBUF *      $001B
0200            PACKT  *      $1A00
0200            SAVX   *      $17E9
                 Set input and output to your routines.
                 Input should echo to output and preserve X.
                 Output should preserve X and ACC.
0200            INPUT  *      $1000
0200            OUTPUT *      $1017

                  Enter here from KIM BASIC INPUT call
0200 24 E7      START  BIT    TIMNUM If not time to output a line number
0202 10 14             BPL    INP    Then branch to input call
0204 A5 E8      LINO   LDA    HI     Otherwise output a 4 digit line number to
0206 20 6A 02          JSR    OUTNUM the BASIC input buffer and users display
0209 A5 E9             LDA    LO
020B 20 6A 02          JSR    OUTNUM
020E A9 00             LDAIM  $00    and clear the TIMNUM flag
0210 85 E7             STAZ   TIMNUM
0212 A9 20             LDAIM  $20    Output a space
0214 20 17 10          JSR    OUTPUT
0217 60                RTS           and return to basic
0218 20 00 10   INP    JSR    INPUT  Get user input
021B C9 40             CMPIM  $40
021D D0 0B             BNE    CRQ    If delete input
021F 24 E6             BIT    SEQFLG and seqence flag is set
0221 10 46             BPL    RETURN
0223 A9 FF             LDAIM  $FF    then set TIMNUM flag
0225 85 E7             STAZ   TIMNUM
0227 A9 40             LDAIM  $40    restore delete
0229 60                RTS
022A C9 0D      CRQ    CMPIM  $0D    If CR
022C D0 3B             BNE    RETURN
022E 24 E6             BIT    SEQFLG and sequence flag is set
0230 30 24             BMI    ENDSEQ go see what to do
0232 A5 1B             LDA    BASBUF otherwise look at BASBUF
0234 C9 23             CMPIM  $23     did he input a #?
0236 D0 2F             BNE    CRRET   If not return with a CR
0238 A9 FF             LDAIM  $FF
023A 85 E6             STAZ   SEQFLG If yes set sequence flag
023C A9 0D             LDAIM  $0D     Output a CR
023E 20 17 10          JSR    OUTPUT
0241 A9 0A             LDAIM  $0A     and LF
0243 20 17 10          JSR    OUTPUT
0246 A2 00             LDXIM  $00    Clear HI, LO and SAVX
0248 8E E9 17          STX    SAVX
024B 86 E8             STX    HI
024D 86 E9             STX    LO
024F 20 90 02          JSR    SETNUM  and go set up HI,LO and INCR
0252 A2 00      SETLNO LDXIM  $00
0254 F0 AE             BEQ    LINO   send the first line number and return
0256 E0 08      ENDSEQ CPXIM  $08
0258 30 09             BMI    CLRSEQ  clear SEQFLG if not enough in buffer
025A 20 C5 02          JSR    INCLN  otherwise add increment to line number
025D A9 FF             LDAIM  $FF    and set the TIMNUM flag
025F 85 E7             STAZ   TIMNUM
0261 30 04             BMI    CRRET  and returns with CR in ACC
0263 A9 00      CLRSEQ LDAIM  $00
0265 85 E6             STAZ   SEQFLG
0267 A9 0D      CRRET  LDAIM  $0D
0269 60         RETURN RTS

                 SUBROUTINES FOLLOW
026A 48         OUTNUM PHA           Puts hex byte in ACC in BASIC buffer as
026B 4A                LSRA          2 ACII decimal digits and ehoes to user
026C 4A                LSRA
026D 4A                LSRA
026E 4A                LSRA
026F 20 7D 02          JSR    HXTAS
0272 20 89 02          JSR    PNUM
0275 68                PLA
0276 20 7D 02          JSR    HXTAS
0279 20 89 02          JSR    PNUM
027C 60                RTS
027D 29 0F      HXTAS  ANDIM  $0F    Changes hex character to ASCII
027F C9 0A             CMPIM  $0A
0281 18                CLC
0282 30 02             BMI    HXI
0284 69 07             ADCIM  $07
0286 69 30      HXI    ADCIM  $30
0288 60                RTS
0289 20 17 10   PNUM   JSR    OUTPUT Line number to buffer and output
028C 95 1B             STAAX  BASBUF
028E E8                INX
028F 60                RTS
0290 E8         SETNUM INX           Gets base line number and increment from
0291 B5 1B             LDAAX  BASBUF buffer and places in HI,LO, and INCR
0293 C9 20             CMPIM  $20
0295 F0 F9             BEQ    SETNUM Ignore spaces
```

```
0297 C9 2B              CMPIM $2B    If plus sign (+) then go get increment
0299 F0 18              BEQ   GETINC
029B 20 00 1A           JSR   PACKT  Otherwise convert to ASCII and store
029E A5 E9              LDA   LO
02A0 A0 04              LDYIM $04    Into HI and LO
02A2 0A         ROT     ASLA
02A3 26 E8              ROL   HI
02A5 88                 DEY
02A6 D0 FA              BNE   ROT
02A8 0D E9 17           ORA   SAVX
02AB 85 E9              STA   LO
02AD 8C E9 17           STY   SAVX
02B0 4C 90 02           JMP   SETNUM And go look for next one
02B3 E8         GETINC  INX
02B4 B5 1B              LDAAX BASBUF Get increment number
02B6 C9 20              CMPIM $20
02B8 F0 05              BEQ   GOTIT  If blank done
02BA 20 00 1A           JSR   PACKT  Convert to ASCII and leave it in SAVX
02BD F0 F4              BEQ   GETINC Go get next one
02BF AD E9 17   GOTIT   LDA   SAVX
02C2 85 E5              STAZ  INCR   put it in INCR and return
02C4 60                 RTS
02C5 A5 E5      INCLN   LDAZ  INCR   Add increment amount to line number
02C7 18                 CLC
02C8 F8                 SED
02C9 65 E9              ADC   LO
02CB 85 E9              STA   LO
02CD A5 E8              LDA   HI
02CF 69 00              ADCIM $00
02D1 85 E8              STA   HI
02D3 D8                 CLD
02D4 60                 RTS
```

A NEW COMMAND FOR BASIC

Dick Grabowski
HDE INC.

To implement the "GET" command in KIM BASIC by Microsoft, chamge the following:

at location $2AEA-$2AEC enter A0 00 A2 1A

The "GET" command allows terminal input and test of a single character without the need to enter the "RETURN" key. One example is in terminal use where you want to hold some material on the screen until the user signals completion:

```
1000 PRINT:PRINT "ENTER SPACE WHEN READY";:GET A$:
     IF A$<>" " THEN 1000
```

This will repeat the prompt "ENTER SPACE WHEN READY" until a space is entered, then fall through to the next program step. Of course, many other uses exist.

GET A ---returns the numeric value 0-9
GET A$---returns the alpha value A-Z

EDITORS ADDITION —
Bob Kurtz of Micro-Z mentioned that he got the "GET" command working by changing location $2AEE from $D0 to $F0.

The GET command will simplify the user interface considerably.

PRODUCT REVIEW

by the editor

Review of Harvey Hermans Basic Enhancement Package as mentioned in Issue #13 p. 12.

Besides what I mentioned in issue #13 you also get a method for using KIMs ST key for stopping the program in a controlled manner, the ability to append Basic programs and subroutines from cassette tape, and a fix for a bug in Basics cassette save routine (also see Herr Schillings letter in the Basic section of this issue for the same fix).

For 15 bucks you get five pages of info. Four pages of explanations and sample printouts of the Basic mods and the fifth page is a hexdump of the mods.

There's no doubt that this is a useful package of info for the Basic user. My only complaints are that the price is a little steep ($7.50 would have been more like it) and the source codes for the mods is not included (Mr. Herman indicated he would sell the source code for an additional 15 bucks).

The "real-time clock" is actually a "tick" counter and not a time of day clock. Could be useful in games etc.

PRODUCT ANNOUNCEMENT

Bob Kurtz of Micro-Z has announced that he is making available his Basic mods which add Hypertape and most importantly the ability to save and load Basic data as well as programs. (This is one of the shortcomings of the 6502 version of Microsoft Basic. It's hard to believe that this ability wouldn't be part of the original program specs.) The ability to save data is a very important one as it enables one to maintain and update data files of such things as income info for tax time, scores for the plant bowling league, handicaps for the local golf course, maintaining your check book, etc etc.

The package also includes a 60 + page Basic manual which explains each command in detail and includes many programmed examples. I have the manual and can attest to its completeness. The only thing this manual doesn't include is any Zero-page usage data as was found in the Basic documentation included with the Basic from Johnson Computer.

Anyhow, Micro-Z is asking $35 for their Basic enhancement package and this uncludes the manual. Contact Micro-Z Co, Box 2426, Rolling Hills, Ca 90274.

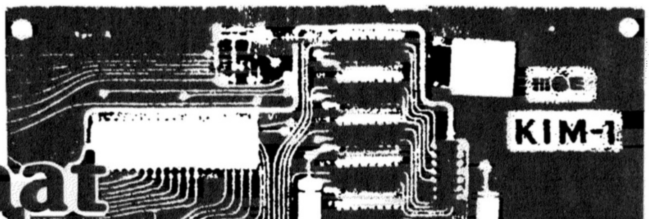
BASIC 'USR' FUNCTION INFO

C. Kingston
6 Surrey Close
White Plains NY 10607

Microsoft BASIC users should note that "AYINT" does not work as described in the instructions for Microsoft BASIC (at least in my version of it.) The USR argument is returned to $00B1(HI) and $00B2(LO), and not to Y and A. After having some problems with the USR statement, I wrote to Microsoft and they repsonded with the above information (apparently not having had this brought to their attention before according to their letter). You should check your version out to see what it does, as newer ones may have this changed either in the program or the instructions. This is applicable for the KB-9 version. I cannot verify that the following work, but Microsoft wrote that the return is to AB(LO) and AC(HI) in KB-6, and B4(LO) and B5(HI) in the ROM version. I would check the HI-LO arrangement carefully, since they had them reversed for the KB-9 version; they work as stated above in my BASIC.

FROM THE EDITOR: The 'USR' function in Microsoft Basic is a kluge at best. Does anyone know where the 'USR' routine is located in Basic so we can make it work right. Tiny Basic has a much better machine language interface.

Do you know what you can do with your KIM?



If not, we have a few suggestions . . .

Two books, whose programs are now available on computer program tapes, will give you numerous suggestions and complete directions for some unique and fun applications for your KIM.

THE FIRST BOOK OF KIM (by Jim Butterfield, Stan Ockers, and Eric Rehnke) is the book all KIM users have been waiting for. In it, you'll find a beginner's guide to the MOS Technology KIM-1 microcomputer as well as an assortment of games and puzzles including Card Dealer, Chess Clock, Horse Race, Lunar Lander, and Music Box. The authors go into detail on how you can expand your KIM from the basic small-but-powerful KIM-1 system to a huge-and-super-powerful machine. Also featured are diagnostic and utility programs for testing both the computer and external equipment (such as cassette recorders), expanding memory, and controlling analog devices. (#5119-0, $9.95) Now available are easy-to-use computer program tapes that feature the 28 recreational and 13 utility programs found in THE FIRST BOOK OF KIM. Tape 1 (#00700) and Tape 2 (#00800) have 14 recreational programs each, and Tape 3 (#00900) has 13 diagnostic and utility programs. All three tapes are $9.95 each.

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT (by Tod Loofbourrow.) "Finally someone has written a book on robot building with microprocessor/microcomputers and a good one at that! . . . a gold mine of useful information on interfacing microcomputers to the real world — and beyond." *Computer Dealer.* You'll experience the thrill of creating an intelligence other than human when you see how "Mike" grows from totally under your control, to seeing and feeling his environment, to responding to voice commands. This book details that creation by giving step-by-step directions for building a robot that is controlled by a KIM-1. (#5681-8, $7.95). The five complete control programs for a robot are clearly documented in the book and are available on a computer program tape. (#00100, $14.95)

#1 in personal computing books!

Books and Tapes are available at your local computer store!

Hayden Book Company, Inc. 50 Essex Street, Rochelle Park, NJ 07662

focal

Lots of neat mods are in store for FOCAL. We're going to add a cassette save & load facility, a Basic-like data statement, output to KIM's seven segment display, the ability to handle arrays of strings, an improved print command, a machine language subroutine call and a few minor fixits and speed-up mods.

Before we do all this, however, we need some room. The present size of the Aresco V3D is about 6K so let's stretch it out to an even 8K and give ourselves a little breathing room. If you examine the listings (love them listings!), you'll notice that the user program must start right after Focal because of line number 00.00 at $35EB.

(One problem: all these mods pertain to V3D which is distributed by Aresco and not necessarily to FCL-65E which is distributed by 6502 Program Exchange. The symbolic addressing info might pertain to FCL-65E but since I don't have a listing of FCL-65E, I can't be sure. FCL-65E might be an updated version of V3D but I can't be sure).

Extend V3D FOCAL to 8K by moving $35EB through $360A to $3FE0-$3FFF. This moves the line 0.0 startup message to the top of the 8K block that will be used by FOCAL. Some zero page pointers must also be changed to allow for the above mod.

```
change:   TEXTBEG $002F   FROM $EB TO $E0
                  $0030   FROM $35 TO $3F
          PBADR   $0031   FROM $09 TO $FE
                  $0032   FROM $36 TO $3F
          VARBEG  $003E   FROM $0A TO $FF
                  $003F   FROM $36 TO $3F
          VARST   $0040   FROM $0A TO $FF
                  $0041   FROM $36 TO $3F
          VAREND  $0042   FROM $0A TO $FF
                  $0043   FROM $36 TO $3F
          PDLIST  $0053   NO CHANGE
                  $0054   FROM $3F TO $5F
```

This is the FOCAL pushdown stack and should be set to some convenient page up out of the way of FOCAL programs. $5F assumes a 16K system.

Another thing that must be improved is the way FOCAL sets up zero-page. Actually, it doesn't. I really can't understand why the implementers overlooked this problem. Oh well...it's easy to fix. At $3F00 add the following zero page intialization routine. $3F00 will become the new cold start address.

```
          ZPAGE  = $0
          ZSTORE = $3F10
          LENGTH = $BF        ; NUMBER OF BYTES
          STARTF = $2000
          *=$3F00
CSTART    LDX #0             ; INITS THE LOOP COUNTER
ZLOOP     LDA ZSTORE,X       ; START MOVING DATA
          STA ZPAGE,X
          INX
          CPX #LENGTH+1
          BNE ZLOOP
          JMP STARTF         ;   PAGE IS SET UP
                             ; GO TO FOCAL
```

Ok now, we have stretched out FOCAL to 8K and added a Z page initialization routine. What next? We'll start adding mods from $35EB-$3EFF.

More next time.......

tiny basic

Oops! In issue #13, I left out the mod that must be made to the IL at $0A26. Here it is:

```
0A26  1E  NX  NX on REM instead of NS.
```

In the next issue, we'll be presenting a very comprehensive string capability for TB as well as a cassette save and load ability. (I ran out of room in this issue). Must be a good number of Tiny Basic users out there. Have you done anything neat with TB? Let us know.

forth

Nothing new to report here except that by issue #15 or #16 I hope to be announcing availability of FORTH for KIM. Preliminary versions are actually operational at this point, but documentation has to be written and other details have to be worked out. This version of FORTH was written to conform to the implementation info presented in the Caltech FORTH manual. A complete source listing will be available.

Beware - just because it's called FORTH don't mean it really is. Before you purchase any package called "FORTH" make sure it conforms to the "international standard" presented in #13.

xplo

Ready for something new? Take a look at XPLO. This is a block structured compiling language that is quite a bit different from BASIC or FOCAL and more along the lines of a subset of ALGOL or PASCAL. An article on XPLO appeared in Kilobaud (Feb 79 p. 24) which should enlighten you on the ins and outs of this new addition to the 6502 users arsenal.

I purchased XPLO from the 6502 Program Exchange, 2920 Moana, Reno NV 89509. Get their catalog for $1. Lots of good software from this group Check them out.

Here's a sample HILO program written in XPLO to give you an idea what it looks like.

```
'CODE' CRLF=9,RANDOM=1,INPUT=10,TEXT=12;
'INTEGER'  GUESS,NUMBER,INCORRECT,TRY;
'PROCEDURE'     MAKEANUMBER;
'BEGIN'
    NUMBER:=RANDOM(100);
'END';
'PROCEDURE'     INPUTGUESS;
'BEGIN'
        GUESS:=INPUT(0);
'END';


'PROCEDURE'     TESTGUESS;
'BEGIN'
    'IF' NUMBER=GUESS 'THEN'
    'BEGIN'
        TEXT(0,"CORRECT!!");
        TRY:=1;
    'END'
    'ELSE'
        'IF' NUMBER<GUESS 'THEN'
        TEXT(0,"TOO HIGH")

        'ELSE' TEXT(0,"TOO LOW");
CRLF(0);
'END';


'BEGIN'
INCORRECT:=0;
TRY:=INCORRECT;
MAKEANUMBER;
'WHILE' TRY=INCORRECT 'DO'
          'BEGIN'
                    TEXT(0,"GUESS ");
                    INPUTGUESS;
                    TESTGUESS;
          'END';
'END'
```

The 6502 Program Exchange also offers a very powerful text editor which is based on the DEC TECO editor. TEC65 is a line oriented (no line numbers) and allows for some very complex editing editing macros. For instance, you could conceivably convert an assembly source file from one assembler format to another. TEC65 includes a cassette operating system which lets you save and load text files.

I've had this editing language running on my machine and am quite impressed with its power. Check it out!

sym!

ACCESSING THE SYM DISPLAYS

A. M. Mackay
600 Sixth Avenue West
Owen Sound, Ontario

I got my new SYM-1 a couple of weeks ago, and I love it. But there's a lot of work to be done-it's not that similar to KIM.

Outputting on the display is a lot different-instead of using F9, FA and FB, you have to treat each 7-segment display as a unit, and get it into DISBUF at A640 (left display) through A645 (right display). But since DISBUF is in write-protected RAM, you have to call ACCESS at 8B86 first to un-write the RAM.

I've enclosed a little SYM-1 program to output the characters, and shown the segment coding. The program as written will display squirrely characters as indicated. Any character can be displayed by changing the coding in 021A through 021F, using the indicated coding, and others can be found starting at location 8C29 in the monitor.

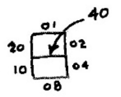
This program may help novices like myself to get the feel of SYM. I haven't had time to figure out the counter yet.

Maybe some bright USER can come up with a way to hook up Don Lancaster's TVT 6 5/8 to a SYM.

Sorry that Jack Cowan had trouble with his Solid State 4K board. Mine worked the first time with my KIM (no bad chips), and when I plugged it into my SYM, after changing the addresses, it also worked perfectly.

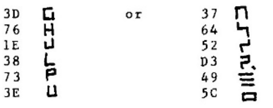
```
ACCESS   =  $8B86
DISBUF   =  $A640-45
SCAND    =  $8906

0200  A2 05                LDX   @ 05
0200  20 86 8B  GETCH      JSR   ACCESS
0205  BD 14 02             LDA   TABLE,X
0208  9D 40 A6             STA   DISBUF,X
020B  CA                   DEX
020C  10 F7                BPL   GETCH
020E  20 06 89  DISPL      JSR   SCAND
0211  4C 0E 02             JMP   DISPL
0214  79        TABLE     .BYTE  'ESCAPE'
0215  6D
0216  39
0217  77
0218  73
0219  79
```



Blank = 00, Decimal = 80
SEGMENT CODE

```
3D        or       37
76                 64
1E                 52
38                 D3
73                 49
3E                 5C
```



Other character codes start at Mon. address $8C29.
To change display change coding in table 0214-0219.

SYM NOTES & KIM-4 COMPATIBILITY

C. Kingston
6 Surrey Close
White Plains NY 10607

I note that some information on the SYM is being included in the 'USER NOTES'. The SYM looked like a reasonable way to improve a KIM based system so I looked into this. The Synertek liturature states that the SYM is usable with any KIM based motherboard. To check, I wrote Synertek and asked specifically if it would work with the KIM-4. They replied that it would. So I got one and tried it, and can report that it does not work with the KIM-4. The trouble was tracked down to the fact that the KIM-4 data buffers are enabled at the wrong times. Thus the KIM-4 has to be altered for proper decoding. After cooling down somewhat about this development, I looked into the problem and came up with a solution, but not an optimal one.

The following alteration will result in the KIM-4 being enabled for all addresses below $8000, and disabled for all above (and including) $8000. Since the SYM in its full glory will utilize almost all of the addresses above $8000, this scheme makes sense. However, if you want to fit in some RAM in the unused high memory positions, you will have to resort to more extensive surgery on the KIM-4. The mod here requires that the lower 1K of memory be on the KIM-4, which makes the RAM on the SYM unnecessary. A simple change will disable the low 1K ($0000-$03FF) on the KIM-4 and allow its use on the SYM. However, since this makes filling in the rest of the first 8K somewhat awkward, I prefer to ignore the SYM RAM (which should be removed, especially if either set of RAM's on the same address can be write protected.)

First of all, you will have to remove chip U5 of the KIM-4 (the 7423) and replace it with a socket. Since you will in all probability destroy U5 in this process, be sure you get another 7423 before starting. Note that some pins make their connection at the top of the board only. If a tiny bit of solder is placed on these pins at the base before putting the socket on the board, it will melt and make contact if you heat the pin adequately. Replacing the 7423 into the socket will leave the KIM-4 in its original state for the KIM.

Now you will connect a 16 pin dip header to a 14 pin socket using the following connections:

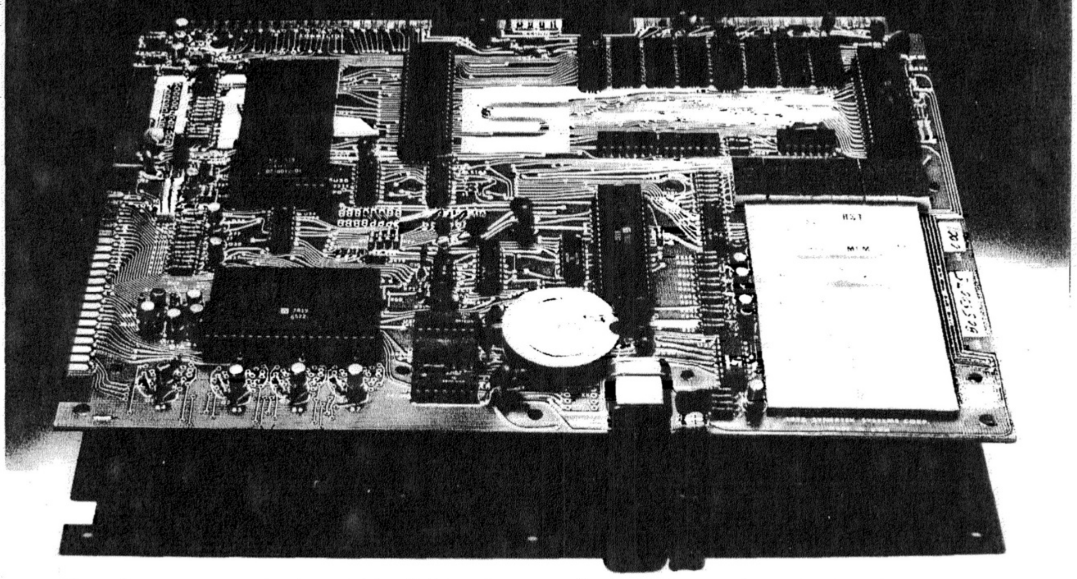
16 PIN (TOP)	14 PIN (BOT)
4	9
6	12
7	8
8 GND	7 GND
9	3
10	2
12	1
16 VCC	14 VCC

Now connect pins 10 and 11 together on the 14 pin socket, and do the same with pins 13 and 14. Be sure to use covered wire as they will have to cross over each other. I made my connections about ½ inch long so that the 14 pin socket sets just above the 16 pin dip header. Using solid hookup wire makes for a fairly firm package. Being careful to observe the location of pin 1, put a 7400 (or 74LS00) into the 14 pin socket. Now plug the converter into the socket for U5 in place of the 7423. The KIM-4 is now enabled for the lower 32K and disabled for the top 32K. The SYM should now work when plugged in, but not the KIM.

To disable the KIM-4 for the first 1K, do not connect pins 13 and 14 on the 14 pin socket together. Instead, run a connection from K0 on the applications connector (pin A-B) to pin 13 of the 14 pin socket. Note that there is an unused nand gate in the 4700 that can be used for additional coding if desired. (The info in this paragraph has not actually been tried yet).

Having made the above conversion, I was then in a position to determine whether or not KIM programs that use TTY or CRT I/O could be successfully transferred and used after changing the I/O vectors and/or JSR's. This led to the next problem. Transferring programs to the SYM system from the KIM system sounds easy since SYM has a KIM format tape input program. But it ain't easy. First of all, as noted in the Feb ('79) MICRO, the SYM stops when it detects a '2F', thinking it is the end of file marker. Those '2F' bytes that are not EOF's have to be changed to something else before transfer, and then changed back afterwards. (I intend writing a short program that will do this in the near future.) The next problem is that SYM will not read tapes made with Hypertape; you must muse the KIM monitor speed. I suppose that a program could be written to read Hypertape, and wonder why Synertek didn't make theirs flexible enough to do this. (If I read the monitor proram correctly - and this is not easy to do because of all the branches and jumps it uses - the number of pulses required per bit is programmed into the monitor and cannot be changed.)

SYM-1.
Finally, a dependable microcomputer board.



In performance. In quality. In availability. OEMs, educators, engineers, hobbyists, students, industrial users: Our Versatile Interface Module, SYM-1, is a fully-assembled, tested and warranted microcomputer board that's a true single-board computer, complete with keyboard and display. All you do is provide a +5V power supply and SYM-1 gives you the rest—and that includes fast delivery and superior quality.

Key features include:
- Hardware compatibility with KIM-1 (MOS Technology) products.
- Standard interfaces include audio cassette with remote control; both 8 bytes/second (KIM) and 185 bytes/second (SYM-1) cassette formats; TTY and RS232; system expansion bus; TV/KB expansion board interface; four I/O buffers; and an oscilloscope single-line display.
- 28 double-function keypad with audio response.
- 4K byte ROM resident SUPERMON monitor including over 30 standard monitor functions and user expandable.
- Three ROM/EPROM expansion sockets for up to 24K bytes total program size.
- 1K bytes 2114 static RAM, expandable to 4K bytes on-board and more off-board.
- 50 I/O lines expandable to 70.
- Single +5V power requirements.
- Priced attractively in single unit quantities; available without keyboard/display, with OEM discounts for larger quantities.

Synertek Systems Corporation.

150-160 S. Wolfe Road, Sunnyvale, California 94086
(408) 988-5690.

To place your order now, contact your local area distributor or dealer.

OEM Distributors
Kierulff Electronics
Sterling Electronics (Seattle only)
Zeus Components
Century/Bell
Lionex
Hallmark
Intermark Electronics
Quality Components
Technico
General Radio
Western Microtechnology
Future Electronics
Alliance Electronics
Arrow Electronics

Personal Computer Dealers
Newman Computer Exchange
Ann Arbor, Michigan

Technico
Columbia, Maryland
Computerland
Mayfield Heights, Ohio
RNB Enterprises
King of Prussia, Pennsylvania
Computer Shop
Cambridge, Massachusetts
Computer Cash
Anchorage, Alaska

Ancrona
Culver City, California
General Radio
Camden, New Jersey
Advanced Computer Products
Santa Ana, California
Computer Components
Van Nuys, California
Alltronics
San Jose, California

I finally managed to transfer Tiny Basic to the SYM system, and after changing the I/O vectors, found that it worked. One point that I noted was that the SYM monitor converts lower case input into upper case, but this looks like it would not be too hard to get around.

Another point of difference in the KIM and SYM. I have found the KIM tape I/O to be extremely reliable and not critical in the recorder used or the settings used. For the short bit that I have used the SYM, I have found it to be extremely sensitive to the recorder volume and tone settings, particularly for the high speed format. It was thoughtful of Synertek to provide a visual means of setting the volume and tone controls for proper reading of the tape (using a sync tape). I guess one must decide which is better - fast tape I/O but critical settings, or slower but surer tape I/O.

ADDENDUM

Since writing the above I have noted another problem in using the SYM with the KIM-4. KIM outputs K1,2,3,4 are wire-or'd together on the KIM-4 and pulled up with a resistor (R2). The equivalent lines on the SYM are not open collector lines and probably should not be wire or'd. No harm came to the 74LS138 on the SYM during the short time I used it without noticing this (at least no noticeable harm). These four lines can be separated on the KIM-4 by cutting the traces between them (where the connectors are soldered to the KIM-4 board). K4 (A-F) can be disconnected from R2 by removing the through plating where the line transfers to the bottom of the board. A four position dip switch could be hooked up to connect or disconnect the lines.

WUMPUS & MUSIC BOX MODS FOR SYM

Jim Adams
17272 Dorset
Southfield, Mi 48075

SYM Users: Make the following modifications to Stan Ockers' WUMPUS program in The First Book Of KIM to use it on your machine.

LOC	FROM	TO	
35C	E7	29	
35D	1F	8C	
365	E7	29	
366	1F	8C	
376	14	47	ASCII G (GO key) to pitch Gas
2A6	06	04	
2A7	17	A4	
2E1	E7	29	
2E2	1F	8C	

Replace 200 thru 257 with

```
200  84 DE 85 DD 20 86 8B A0 05 B1 DD 49 80 C9 80 F0
210  1F 99 40 A6 88 10 F2 A2 0A 86 DB A9 52 8D 1F A4
220  20 06 89 2C 06 A4 10 F8 C6 DB D0 EF E6 DD D0 D7
230  60
```

Replace 258 thru 271 with

```
258  20 AF 88 C9 47 F0 05 20 75 82 B0 F4 60
```

Make the following modifications to Jim Butterfield's MUSIC BOX program in The First Book Of KIM to play music on your on board speaker.

LOC	FROM	TO
20B	BF	0D
20C	8D	20
20D	43	A5
20E	17	89
219	00	60
24C	A7	06
255	27	08
270	42	02
271	17	A4

ATTENTION "VIM" AND "AIM-65" USERS!!! THE SAN FERNANDO VALLEY KIM-1 USERS CLUB IS EXPANDING ITS MEMBERSHIP TO INCLUDE THESE TWO NEW AND EXCITING MICROCOMPUTER SYSTEMS WE MEET AT 7:30 PM ON THE SECOND WEDNESDAY OF THE MONTH AT 20224 COHASSET #16, CANOGA PARK, CA 91306 CALL JIM ZUBER AT (213) 341-1610 IF YOU HAVE ANY QUESTIONS.

aim!

MANUAL CORRECTIONS

Jody Nelis K3JZD
132 Autumn Drive
Trafford Pa 15085

I have had a Rockwell AIM 65 for three weeks now and I can only say one thing: Fantastic machine!!!

The AIM is following a one year session with a KIM-1. The 8K monitor in the AIM takes care of a lot of the things that the KIM monitor needed additional software to handle.

The text editor is slick especially for we typists who make a lot of mistakes. Entering a program in mnemonics is a step up from all Hex Op Codes.

The user's guide shipped with the AIM was hurridly put together since it held up shipment of the hardware. To help others who may have an AIM 65, I'm passing along several items that I believe to be incorrect in the user's guide (October 1978 issue).

Page 2-19 A step is missing in the program entry for this example. Between the "AND #0F" and the "BRK" there should be a "STA #41".

Page 2-25 At the top of the page, the display register format should read: PC P A X Y S

Page 3-20 There is a problem with the form given for the (indirect),Y addressing mode. See the separate sheet that discusses that subject.

Page 3-23 Under using the K command:
1. Type K. AIM 65 will repsond with: <K>*=
2. Enter the starting address in hexadecimal. AIM 65 will respond with: <K>*=0300
3. Type return. AIM 65 will respond with: /
4. Specify the number of instructions.. ETC.

Page 9-11 The syn test pattern program has multiple errors in it. This is liable to have a lot of people wearing out VR1 or spending a lot of time trouble-shooting an OK tape recorder. I have included a corrected program disassembled from the AIM 65 on a separate sheet.

Page 11-3 The Olivetti type no. 295933R35 thermal paper that is refered to can't be located in the Pittsburgh area using that number. In fact all of the Olivetti Dealers I talked to were unable to come up with any 2½ inch wide paper at all. I have found that Texas Instruments #TP-27225 thermal paper is the right size and price (3 rolls for $3.69) and is available anywhere that TI calculators are sold. I haven't tried the SEARS paper yet but the catalog at least says it exists. In a pinch, Radio Shack sells 3 smaller rolls for $2.79 that will work also (Cat. No. 65-706).

Page A-1 There is a problem with the form given for the (indirect),Y addressing mode. See the separate sheet that discusses that subject.

Page K-10 The disassembly listing on this page and the next have been interchanged.

Hopefully these corrections will aid any other new AIM owners get aquainted with their machine.

more...

KIM™ BUS EXPANSION!

AIM™, VIM™, (SYM)™, KIM™ OWNERS (and any other KIM™ bus users) buy the best 8K board available anywhere:

GRAND OPENING SPECIAL!

HDE 8K RAM-$169! 3 for $465!

Industrial/commercial grade quality: 100 hour high temp burn-in: low power: KIM bus compatible pin for pin: super quality & reliability at below S-100 prices (COMMERICALLY rated S-100 boards cost 25-75% more). When you expand your system, expand with the bus optimized for 8 bit CPU's, the Commodore/Mos Technology 22/44 pin KIM bus, now supported by Synertek, MTU, Rockwell International, Problem Solver Systems, HDE, the Computerist, RNB, and others!

KIM-1 computer $179.00: KIM-4 Motherboard $119: power supply for KIM-1 alone—$45: enclosure for KIM-1 alone $29: HDE prototype board with regulator, heatsink, switch address & decoding logic included $49.50: book "The First Book of KIM" $9.95: book "Programming a Microcomputer: 6502" $8.95: SPECIAL PACKAGE DEAL; KIM-1, power supply, BOTH books listed above, ALL for $208!

HDE FILE ORIENTED DISK SYSTEM (FODS) FOR KIM BUS COMPUTERS Make your KIM (or relative) the best 6502 development system available at any price. Expand with HDE's full size floppy system with FODS/Editor/Assembler. 2 pass assembler, powerful editor compatible with ARESCO files KIM bus interface card: fast 6502 controller handles data transfer at maximum IBM single density speed for excellent reliability: power supply for 4 drives: patches to Johnson Computer/Microsoft BASIC. 45 day delivery. Single drive—$1995 dual drive $2750

Shipping extra unless order prepaid with cashier's check ALL items assembled, tested, guaranteed at least 90 days.

PLAINSMAN MICRO SYSTEMS (div. 5C Corporation)
P.O. Box 1712, Auburn, Al. 36830: (205)745-7735
3803 Pepperell Parkway, Opelika
(1-800-633-8724) Continental U.S. except Al.

Dealers for OSI, COMMODORE, COMPUCOLOR, ALTOS

VISA

HDE inc.

HUDSON DIGITAL ELECTRONICS, INC.

BOX 120, ALLAMUCHY, N.J. 07820 • 201-362-6574

KIM-1 PRODUCTS FROM HDE, INC.

DM-816-M8 8K STATIC RAM MEMORY

This is the finest memory board available for the KIM-1 at any price. Commercial/Industrial quality. All boards are continuously operated and tested for a minimum of 100 hours prior to release. Full 6 month parts labor warranty.

DM-816-DI1 8" FLEXIBLE DISK SYSTEM

Available in single and dual drive versions. Includes interface card, power-supply, Sykes controller and drive, cables and manual. File Oriented Disk System software with HDE text editor.

DM-816-MD1 5" FLEXIBLE DISK SYSTEM

Single and dual drive versions include interface/controller, power supply, Shugart drive, cables and manual. Advanced version of FODS software with HDE text editor. Latest addition to HDE peripheral product line.

DM-816-CC15 MOTHER BOARD

A professional mother board for the KIM-1. All KIM-1 functions remoted, includes power on reset. 15 connectors. Provision for Centronics printer interface. Card cage and cabinet configurations available.

DM-816-UB1 PROTOTYPE CARD

Designed for ease of special applications development. Handles up to 40 pin dips.

HDE ASSEMBLER

An advanced, two pass assembler using 6502 cross-assembler mnemonics. Free form, line oriented entry. Directives include: .OPTION, .BYTE, .WORD, .FILE, .OFFSET, .END. Output options include: LIST, NOLIST, SYMBOLS, NOSYMBOLS, GENERATE, NOGENERATE, ERRORS, NOERRORS, TAB, NOTAB. Assemble from single or multiple source files. Place source, object and symbol table anywhere in memory. Automatic paging with header and page number. User's manual. Approximately 4K. Loads at 2000 or E000. Specify on order.

HDE TEXT OUTPUT PROCESSING SYSTEM (TOPS)

A comprehensive output processor, including left, right and full justification, variable page length, page numbering (Arabic or U/C and L/C Roman), page titling, string constants, leading and trailing edge tabbing, field sequence modification, selective repeat, selective page output and much more. Over 30 commands to format and control output of letters, documents, manuscripts. User's manual. Approximately 4K. Loads at 2100 or E100. Specify on order.

HDE DYNAMIC DEBUGGING TOOL (DDT)

Built in assembler/disassembler coupled with program controlled single step and dynamic breakpoint entry/deletion facilitates rapid isolation, identification and correction of programs under development. Keystrokes minimized with single letter, unshifted commands and optional arguments. User's manual. Approximately 2K. Loads at 2000 or E000. Specify on order.

HDE COMPREHENSIVE MEMORY TEST (CMT)

Eight separate diagnostic routines test for a variety of memory problems. Each diagnostic, the sequence of execution, the number of passes and halt/continue on error is selected by the user on call-up. Tests include pattern entry and recall, walking bit, data-address interaction, access time and cross talk, simulated cassette load, slow leaks. Suitable for static and dynamic ram. User's manual. Approximately 3K. Loads at 2000 or E000. Specify on order.

HDE TEXT EDITOR (TED)

Complete, line oriented text editor accepts upper or lower case commands. Functions include line edit, line move, line delete, block delete, resequence, append, list, print, locate, set, scratch, automatic/semi-automatic line numbering, lastcommand recall, job command. This editor is supplied with all HDE Disk Systems. User's Manual. Approximately 4K. Loads at 2000 or E000. Specify on order.

ALL PROGRAMS ARE AVAILABLE FOR LOCATIONS OTHER THAN THOSE SPECIFIED AT ADDITIONAL CHARGE.

	Disk-Note A	Cassette-Note B	Manual Only	Note C
HDE Assembler	$ 75.00	$ 80.00	$ 5.00	$25.00
HDE Text Output Processing System (TOPS)	135.00	142.50	10.00	15.00
HDE Dynamic Debugging Tool (DDT)	65.00	68.50	5.00	5.00
HDE Comprehensive Memory Test (CMT)	65.00	68.50	3.00	5.00
HDE Text Editor (TED)	N/C	50.00	5.00	15.00

Note A. Media charge $8.00 additional per order. Save by combining orders.
Note B. Cassette versions available 2nd qtr. 1979.
Note C. Additional charge for object assembled to other than specified locations.

ORDER DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER	PLAINSMAN MICROSYSTEMS	ARESCO
Box 523	Box 1712	P.O. Box 43
Medina, Ohio 44256	Auburn, Ala. 36830	Audubon, Pa. 19407
216-725-4560	800-633-8724	215-631-9052

AIM 65 User's Guide October 1978 edition page 9-11

Corrected SYN test pattern program

```
END

/03
 0300 20 JSR F21D
 0303 20 JSR F24A
 0306 4C JMP 0303

<K>*=0310
/11
 0310 A2 LDX #00
 0312 A9 LDA #CE
 0314 20 JSR EF7B
 0317 20 JSR EDEA
 031A A2 LDX #00
 031C A9 LDA #D9
 031E 20 JSR EF7B
 0321 20 JSR EE29
 0324 C9 CMP #16
 0326 F0 BEQ 0321
 0328 D0 BNE 0310
```

CONTRADICTIONS & CONFUSION IN THE TABLES OF ADDRESSING MODE FORMATS FOR USE WITH THE AIM 65 MNEMONIC ENTRY MODE

On the AIM 65 Summary Card:

The operand format given for the (indirect),Y addressing mode is incorrect. Both the (HH,Y and the (HH,Y) formats, when entered, end up being decoded as an (indirect,X) opcode. Needless to say, this bombs a program badly!!

In the AIM 65 User's Guide on page A-1:

The same comments as above apply.

In the AIM 65 User's Guide on page 3-20:

One of the operand formats given for the (indirect),Y addressing mode is correct here. The (HH)Y works fine. The (HH,Y) format given here is no good as noted above.

SUMMARY:

The (HH)y operand format on page 3-20 is the only one given that runs properly. By experimentation, I have also found that (HH),Y also works and should be listed as the alternate for those who like it longer but correctly written.

I also noted that all three of the sources of information on the mnemonic instruction operand listed above have an addressing mode listed as (absolute indirect). I'm no expert on the 6502, but I know of no such addressing mode!!

(EDITORS NOTE: What about JUMP INDIRECT?)

VIDEO & TVT-6

POLYMORPHICS VIDEO/KIM INTERFACE

by Mike Firth
104 N St Mary
Dallas Tx 75214

I am using a Polymorphics Video Interface with my KIM-1. My purpose here is to describe what I did to make it work (besides spend money), which was simpler (except for my mistakes) than I thought it would be. Some of what I have to say will be applicable to using an S100 memory board, since the VDM is memory mapped.

I selected the Polymorphics board because, at the time, it was the only display that would give me both upper and lower case (which I needed for editing) along with graphics. I felt for my purposes, graphics would be more useful than the reversed background offered on other S100 boards. (Besides I have worked on terminals with white background and the glare bothers me.) I also wanted to have the option of displaying control characters, which SWTC and other TVT's usually don't. I wanted software control.

The graphics on the Polymorphics board consists of using the lower six bits of the word to each control one small square fitted in three rows of two each that fill all the space taken by a letter. That means the graphics fit edge to edge. One letter character (111111) fills a portion of the block, like typing capitals on top of each other, thus ■.

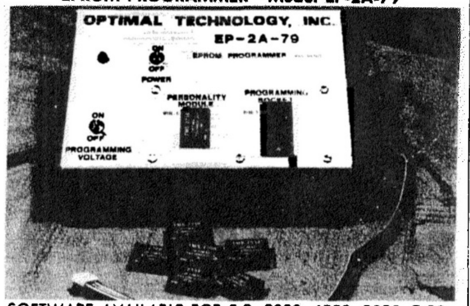
One problem with the graphics is that some strange early decision decided that having bit 7 (most significant bit) set (=1) would be ASCII character, while unset (=0) would be graphics. One could add and remove the bit with software, but I am going to invert the data line to the video board so tests, et.al. are easier.

Any S100 board provides many control lines because of the way the 8080 accesses memory and I/O. The 8080 requires separate data in and data out buslines and also must allow for I/O Ports that are activated with address lines and an a PINT line. That also means that there has to be a control for memory read and another for memory write, which there is. Also, since the 8080 is an early device, really odd blips can appear on the address lines while the CPU is internally working, so there are timing restrictions.

We are fortunate because the address lines stay valid for the entire clock cycle and because the data lines are bi-directional. Since the video board has both in and out buffered, we can just tie the appropriate lines (eg. bit 7 in and bit 7 out) together.

The RAM R/W line from the KIM turns out to be the only line we need to control the access to the video board. IF WE ARE WILLING TO ADD A COUPLE OF JUMPERS ON THE BOARD AND CUT TWO TRACES, we don't need any chips off the board. (What I am about to describe can be done off the board, sort of.) All of the changes take place in a one inch square area on the board.

EPROM PROGRAMMER—Model EP-2A-79



SOFTWARE AVAILABLE FOR F-8, 8080, 6800, 8085, Z-80, 6502, KIM-1, 1802, 2650.
EPROM type is selected by a personality module which plugs into the front of the programmer. Power requirements are 115 VAC, 50/60 HZ at 15 watts. It is supplied with a 36 inch ribbon cable for connecting to microcomputer. Requires 1 ½ I/O ports. Priced at $145 with one set of software, personality modules are shown below.

Part No.	Programs	Price
PM-0	TMS 2708	$15.00
PM-1	2704, 2708	15.00
PM-2	2732	25.00
PM-3	TMS 2716	15.00
PM-4	TMS 2532	25.00
PM-5	TMS 2516, 2716, 2758	15.00

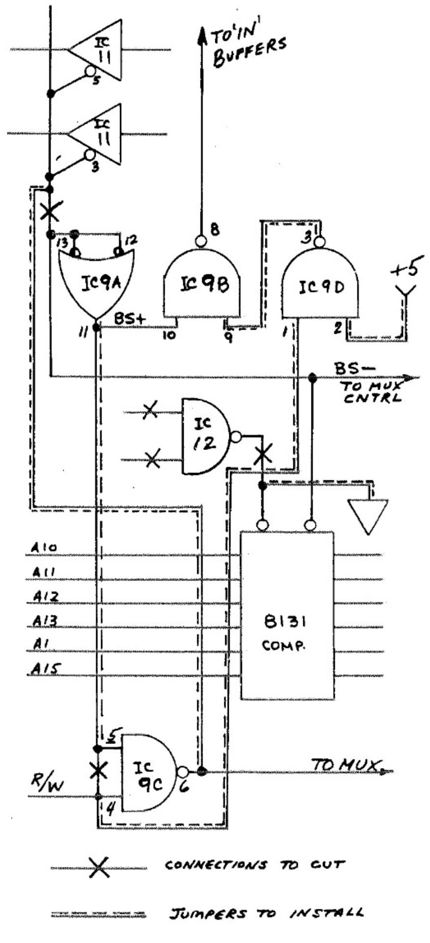
Optimal Technology, Inc.
Blue Wood 127, Earlysville, VA 22936
Phone 804-973-5482

We take advantage of an unused gate in IC9, a 74LS00, and we cut a trace to take advantage of another gate originally used as an invertor. The other trace to cut is the one that controls the data bus from the board to the computer. As wired, it is always connected, except during a write, when it goes tristate. By installing a jumper, it will be tristate except when the video board is actually addressed.

(Editors Note: The I.C. numbers and the pinout refer to an early model of Polymorphics VTI-64 Video Board. I have the Rev 'F' version and had to do some transposing of I.C. numbers and pinouts, but got everything working OK.)

As shown on the circuit, BS- goes low when the proper 1K of memory is addressed. This is inverted through IC9A, then is usually NANDed with 8080 memory write to control the data buffers into the board. We are going to replace MWR+.

We bring in the KIM RAM R/W line on pin 47 to IC9C as usual. But if this is all we did, the inverted signal would messup the video display EVERY TIME WE DID A WRITE OPERATION ANYPLACE. So, we cut the trace to pin 5 if IC9 and run a jumper from pin 11 to pin 5 of IC9 (arn't we intimate). This insures that pin 6 is low only during access to this block. We then install a jumper from IC9 pin 6 to IC8 pin 3, which controls the tristate (and is next to IC9).



MODS TO POLYMORPHIC VTI-64

Two more jumpers remain. One goes from pin 4 of IC9 to pin 1, carrying the RAM R/W signal to the unused gate in IC9 to use it as an invertor. The output of IC9D (pin 3) is jumpered to IC9B (pin 9) to replace the 8080 MWR+ mentioned earlier. Also, pin 2 of IC9 has to be jumpered to +5 (better practice) or to pin 1.

And that is that. Connect the video board to a monitor, plug everything in and wait. The Polymorphics manual suggests aid for the board. If you get a good stable display, write a program that will load a regularly varying bit pattern into successive locations. If the display shows the same character repeated 2 times, or four times, you probably have miswired something and lost a data line or two (I did, two lines were on the wrong pins on the S100 connector). If you can't access a part of the screen, or the program writes over some areas and doesn't touch others, then an address line is buggy. You should note that even if you only have 16 lines 32 characters long (512 characters), the video board takes up 1K of memory, since it is A5 that is ignored. This save rewriting some software if you add the other 32 later.

With KIM, you can use the monitor to load (and read) any location one-by-one. KIM accesses the memory periodically, which you will note as a line on the screen which disappears if you access elsewhere.

TVT-6 NOTES & RAM EXPANSION

by Milan Merhar
697 Boylston St
Brookline Ma 02146

More TVT-6 stuff and another way to fill the lower 4K hole in KIM.

I read the letter in #13 re the TVT-6 and I thought I'd pass along some comments.

My cassette copy of the TVT-6 programs from PAIA were very corrupt. Obviously, they were keyed in, not checked, and recorded. Check the programs against the listings before running.

One listing error for the cursor routine: the contents of $0185 should be $03 rather than $01.

Dennis Chaput's problems with the cursor routine not working can be traced to Don Lancaster's cryptic note at the end of the cursor program listing: "To protect page entry, load 00F3 04, to enable page entry, load 00F3 00".

This initializes the accumulator to 04 or 00 via KIM's monitor at run time and therefore Does Nothing!

Obviously, Lancaster meant to initialize 00F1 to 04 or "initialize the status byte to disable interrupts" to protect the page and initialize 00F1 to 00 or "enable interrupts" to access the cursor routine normally.

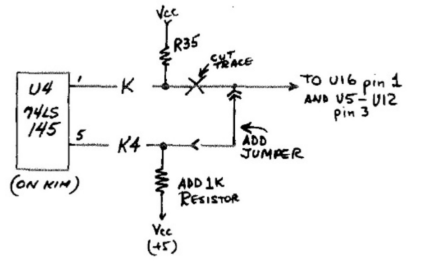
Also note that "Erase to end" and"Sparehook" comments are reversed in the cursor program commentary. ASCII 13 is "erase to end of line", ASCII 12 is "space".

If you're running Tiny Basic or the TVT-6 from low memory, you usually want to expand memory at low addresses rather than start all over again at 2000 hex and up. Most suggestions to squeeze 4K of RAM into the lowest memory space involves fancy bussing of individual chip selects to each 1K of RAM to be added. There's an easier way!

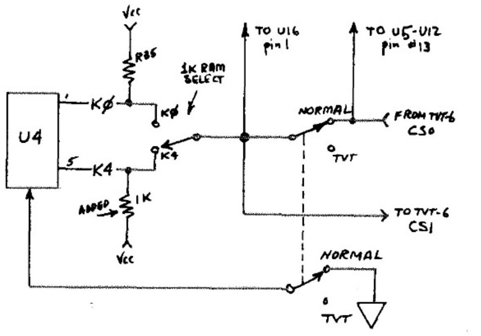
Set your commercially available 4K memory board to fill addresses 0000 hex to 0FFF hex and move KIM's 1K RAM to addresses 1000 hex to 13FF hex.

This gives 5K of contiguous RAM for use with Tiny Basic at 0200 hex or such.

The mod requires a jumper wire and one 1000 ohm resistor to be added.



Please note that "page zero" and the stack are now physically in the 4K RAM board which must be present for the monitor to function. Incorporating this mod to the mods for the TVT-6 gives something like this:



The SPDT switch selects whether the KIM 1K RAM is at 0000 hex to 03FF hex (with the 4K RAM removed!) or at 1000 hex to 13FF hex (with the 4K RAM present)

The DPDT switch is for normal use with the TVT-6 out of its socket.

The TVT-6 can be wired up to the 4K board as discussed by Michael Allen in #13 or can work out of KIM's 1K RAM at 1200 hex to 13FF hex. Patches are simple: The SCAN routines work is written: the cursor is changed by adding 10 hex to the contents of these locations: 0106, 010A, 016E, 0185, 01A0, 01C7, 01DC.

If you are using Tiny Basic, move the cursor routine to 1100 hex and Tiny Basic can have 1½K for user programs from 0B00 hex to 10FF hex. Make sure to initialize Tiny's program space pointers to keep from overwriting your cursor program & display memory in a normal "cold start".

INTERFACING TO THE TVT-II

by John M. Rensberger
1920 NW Milford Way
Seattle Wa 98177

The KIM-1 TTY input and output interface nicely and simply with the CT 1024 TV Typewriter II. However, discovering how to effect this was not easy. MOS Technology supplies no information in their otherwise very extensive literature.

I finally discovered, after 50 hours of searching for a problem in the serial interface and UART of the TVT II, that there is no parity bit, the 8th bit is a 1, and the polarity of the signal as it comes from the transistor interface of Rick Simpson (July User Notes Vol 1) is inverted with respect to the RS232 input requirements of the TVT II serial interface.

Therefore, users attempting to make this interface should use or beware of the following conditions:

1. Omit the inverter in Simpson's circuit.
2. Program the serial interface of the TVT II for NO parity, bit 8=1.
3. I used neither ground nor the -5 volt option on Simpson's circuit. (*EDITORS NOTE: I didn't understand this one*).
4. Lower the resistence of the RS232 input of the TVT serial interface by replacing R-19 (1K) with a 300 resistor.

Knowing the correct bit pattern and that the polarity from the KIM system is correct without inversion should make the interfacing task very simple for anyone wishing to use a TV typewriter as a terminal.

Gary and Lisa Rensberger (ages 14 and 16) have completed a machine language program for KIM that will work with either TVT II or TTY. They would be delighted to hear from users who would like a TIC-TAC-TOE game (700 bytes) in exchange for any other game (listing or KIM compatible tape).

CASSETTE stuff

MAKE A SHORT CASSETTE

Ted Beach - K4MKX
5112 Williamsburg Blvd.
Arlington Va 22207

Regarding your search for C15 tapes, I gave up on that idea long ago. Instead, what I started to do originally was buy "cassette repair kits" sold by Lafayette Radio for about 69 cents each. These are tape housings with a leader from spool to spool. I then spliced in my required length of tape which I cut off of a good quality C60 tape. Radio Shack and Lafayette both have inexpensive (about $4.00) cassette splicing machines.

That drill got a bit expensive, so I started buying "El Cheapo" drugstore tapes (three C30's for $1.29) and discarding the tape from the housings. Works real neat. Now I record the desired number of programs or subroutines on my good C60 tape, run off another foot or so of tape, then clip the tape in the center of the middle opening (where the pressure pads are). Then I pull both ends of the tape out and splice the end coming from the takeup spool to the leader going to the supply spool of El Cheapo cassette (tape already discarded).

A pencil stuck in the hub of the supply spool will quickly rewind the tape into its new housing. At the end, cut the tape from the original leader, splice it onto the leader of the takeup spool of the new housing and reconnect the free end of the C60 tape to its takeup leader. You will eventually run out of takeup leader on your "good" tape, but what the heck - tape is cheap!

As an alternative, you can allow a foot or so of the beginning of the original tape to be used as a leader (mark it with a marker pen and don't start recording until this part is on the takeup spool). This way you can get amximum use from the good tape by sacrificing a foot of it to begin with. When you're through, you will have another empty cassette housing to use, with full length leaders.

CASSETTE DIRECTORY PRINTOUT PROGRAM

...prints your tape directory on your TTY or terminal

Chris McCormack
116 Milburn Lane
East Hills, NY 11577

This program is an expansion of the directory program, written by Jim Butterfield. The advantage of this program is that it will search a whole tape, and output the ID, starting address, and ending address of any program found. Because all of the branches are (EXC 0037) relative, the program is completely relocatable. Program start is at address 005F.

```
0000  D8          TOP     CLD
0001  A9 07               LDA #$07
0003  8D 42 17            STA SBD
0006  20 41 1A    SYN     JSR RDBIT
0009  46 F9               LSR INH
000B  05 F9               ORA INH
```

```
000D  85 F9                   STA INH
000F  C9 16         TST       CMP #'SYN
0011  D0 F3                   BNE SYN
0013  20 24 1A                JSR RDCHT
0016  C6 F9                   DEC INH
0018  10 F5                   BPL TST
001A  C9 2A                   CMP #'*
001C  D0 F1                   BNE TST
001E  A2 FD                   LDX #$FD
0020  20 F3 19      RD        JSR RDBYT
0023  95 FC                   STA POINTH+1,X
0025  E8                      INX
0026  30 F8                   BMI RD
0028  A2 02         MORE      LDX #$02
002A  20 24 1A      SECOND    JSR RDCHT
002D  C9 2F                   CMP #'/
002F  F0 09                   BEQ OUT
0031  CA                      DEX
0032  D0 F6                   BNE SECOND
0034  20 EA 19                JSR INCVEB
0037  4C 28 00                JMP MORE
          (NOTE: MUST BE CHANGED IF RELOCATED)
003A  A5 F9         OUT       LDA INH
003C  20 3B 1E                JSR PRTBYT
003F  20 9E 1E                JSR OUTSP
0042  20 9E 1E                JSR OUTSP
0045  20 1E 1E                JSR PRTPNT
0048  18                      CLC
0049  AD ED 17                LDA VEB+1
004C  65 FA                   ADC POINTL
004E  85 FA                   STA POINTL
0050  AD EE 17                LDA VEB+2
0053  65 FB                   ADC POINTH
0055  85 FB                   STA POINTH
0057  A9 2D                   LDA #'-
0059  20 A0 1E                JSR OUTCH
005C  20 1E 1E                JSR PRTPNT
005F  20 2F 1E      START     JSR CRLF
0062  20 32 19                JSR INTVEB
0065  18                      CLC
0066  90 98                   BCC TOP
```

ZIPTAPE CASSETTE INTERFACE (a review)

Wanna be able to load BASIC in 13 seconds? That's right - 13 seconds for an 8K load (about 12 times faster than HYPERTAPE).(600 bytes/sec).

As you can tell, I'm pretty enthusiastic about the cassette interface from Lew Edwards. So far, this system has proved 100% reliable at 4800 baud. I use a Sankyo ST-50 cassette recorder.

This interface consists of a 2"x2" p.c. board using a single IC (needs 5 v. @ 10 ma.) and 346 bytes of driver software. It uses 3 bits (PA0, PA1, and PA2) of KIM's I/O, and from $0200 to $02A8 and $0300 to $03B2 of KIM's memory.

The documentation, which includes a full source listing of the driver software as well as a schematic of the hardware, comes with enough information to enable the user to get this system running on ANY 6502 system that has 3 bits of a 6530, 6532, 6520 or 6522 available for use.

This system can be operated at 2400 or 3600 baud if your recorder can't handle the full 4800 baud speed.

For $26.50 you get an assembled interface board, a cassette of the software, and full documentation from Lew Edwards, 1451 Hamilton Ave., Trenton, N.J. 08629.

If you just can't swing a floppy-disc, Ziptape will ease the pain.

-ERIC

CASSETTE AVAILABILITY

Are you looking for some high-quality short and medium length cassettes? AB Computers (POB 104, Perkasie, PA 18944 (215)257-8195) has some which come in 5-screw housings and use AGFA tape.

We use these cassettes for software distribution here at USER NOTES and have been quite satisfied with them. Here's the price list:

C10 (5min/side)	10/$6.25
C30 (15min/side)	10/$8.00
plastic housings	10/$1.00

EDITORIAL (continued from inside the cover)

Things we'd like to know: What boards have you used successfully in the KIMSI? What mods did you have to perform to get other boards operational? Has anyone figured out a way of modifying the KIMSI so that the special I/O port of the memory map is moved down into KIMs' 4K "black hole" ($0400 $13FF)?

MANUFACTURERS

Need some ideas for new products? The 44 pin KIMBUS is gaining in popularity now that Rockwell and Synertek have entered the marketplace with products intended to be used with the KIM-4. There's always room in the RAM board market, how 'bout a low-cost dynamic RAM board which can take advantage of the super low-cost 4116 which are being offered for the TRS-80 and APPLE machines. I've seen a set of eight going for as low as $80.00. That's 16K!!! A good 64x16 or 80x24 video board is desperately needed.

How 'bout an EPROM board that can also program the 2708 or 2716. (At this point, the hobbyist is money ahead by sticking to the 2708, as low as $5.00, unless he really needs the single voltage of the Intel 2716.) A combination serial/parallel board using the 6522 and maybe the new Synertek 6551 ACIA would be very popular (2P+2S?).

BASIC INCOMPATIBILITY

Microsoft has written versions of BASIC to run on all major 6502 machines (KIM, APPLE, PET and OSI.) Although, for the most part, a program written for one machine can be run on another machine if they are typed in, a memory image cannot be transferred from one type of machine to another (PET to KIM, for example). The reason? First of all, when you type a BASIC program into your computer (with Microsoft BASIC, anyway) the program is compressed by changing BASIC commands into

CONTINUED ON NEXT PAGE

KIM SOFTWARE ON CASSETTE

We know that you have better things to do with your time than punching hex code into your machine. Because of this, we have made some of the longer programs available on KIM cassette.

These cassettes are original dumps, not copies, made with top quality 5-screw housing cassettes. Thirty seconds of sync characters precede the program to enable you to tune up your recorder or PLL.

Are you AIM & SYM owners interested in having some of these programs available for your machines?

6502 USER NOTES, POB 33093, N. ROYALTON OHIO 44133

OUR PRESENT OFFERINGS INCLUDE:

KIMATH (specify $2000 or $F800 version).....$12.00 (includes errata sheet for manual)

HEXPAWN (from issue #13)....................$5.00

DISASSEMBLER (this issue)...................$5.00

BANNER (this issue).........................$5.00

PAYMENT MUST BE IN U.S. FUNDS. OVERSEAS CUSTOMERS--include $1.00 extra per cassette for extra postage.

"tokens". For example, if you type in 100 PRINT "HI", BASIC would store two bytes for the address, and one byte for PRINT. "HI" would be stored without change. Using one byte "tokens" lets you get a larger BASIC program into smaller areas and could even help them run faster. Only one problem: Each different version of BASIC uses its own unique 'token' identifiers. (Does anyone know why this is so?)

The only way to transfer BASIC programs from one type of 6502 machine to another is to "LIST" the program out to the other computer. In other words, instead of listing the BASIC program to your printer, the output would be "vectored" to a new output routine which would talk to your object computer. The object computer would also be running BASIC and it would expect its "input" from the first computer instead of its own keyboard. Tricky, huh?

The PET, for example, is set up to list a program to a device on its IEEE bus so if I wanted to transfer some BASIC programs from the PET to the KIM, I would hook the KIM up to the PET's IEEE bus and "list" the program to the KIM. Don't forget, KIM needs to have BASIC running and modified so that its input comes from the IEEE bus so that its interface instead of the terminal.

Such a PET to KIM BASIC interface is on my list of projects so it may get done in my lifetime.

<u>32K RAM FOR AROUND $200</u>

Those of you that are running KIM-4, or compatible, motherboards will be happy to hear that I now have in my hot little hands an article on how to build a 32K dynamic RAM board using the 4116 devices which are being used to expand the APPLE computer. (You've seen them in all the mags for $80 - $100 per 16K.) The RAM card contains its own built-in, invisible refresh circuit and can be built on a 4.5x6 size Vector wire wrap card.

According to the author -"In eight months of constant use with a KIM-1 and KIM-4, no problems of any kind have been encountered with the unit. A second unit, built at the end of 1978, also works well."

Sorry, you'll have to wait for the next issue for this one.

announcements and reviews

<u>PRODUCT ANNOUNCEMENT</u>

Tiny Editor/Assembler and Robot

If you only have 4K of expansion and want to assemble programs, (once you start assembling your programs, you'll never go back to hand assembly!) You may want more info on this tiny editor/assembler package. The flyer didn't mention what type of I/O device was supported or where the package was assembled to operate from, but the ROBOT language supports the TVT-6 and needs RAM expansion starting from $0400 so the assembler requirements may be similar.

The single pass assembler overlays the editor in RAM and, except for the zero page references, seems to conform to the MOS definitions.

The price is $23.00 for the user manual, commented source listing ($20.00 without the cassette).

ROBOT is actually an interactive robot control language. The robot that is controlled is currently the cursor on the TVT-6 video board but it looks as if the user could modify this to control a "real" robot or a standard memory mapped video output device. ROBOT needs memory from $0200-$0540 and a TVT-6. I have this package and will be reassembling it to run an my system as soon as I get some time (fat chance!). This looks to be a very interesting package.

Several articles on an 8080 version have been published in Doctor Dobbs Journal.

ROBOT sells for $8.00 and includes a user manual, commented source listing and a cassette ($5.00 without the cassette). It's worth $5.00 just to see how it works!

Contact Michael Allen, 6025 Kimbark, Chicago Ill 60637.

PRODUCT REVIEW of the HDE DISC SYSTEM by the editor

A number of you have asked for details about the HDE full size disc system.

The system is based around the SYKES 8" drive with the 6502 based intelligent controller.

This drive is soft sectored, IBM compatible, and single density which lets you store about a quarter megabyte of data on a disc.

The system software, called FODS (File Oriented Disc System), manages sequential files on the disc much the same way files are written on magnetic tape - one after another. When a file is deleted, from a sequentially managed file system, the space that the file occupied is not immediately reallocated, as in some disc operating systems. As it turns out, this can be an advantage as well as a disadvantage since deleted files on the FODS system can be recovered after the file has been deleted. (This has saved my sanity more than once!) Of course when you want to recover some of the disc space taken up by a number of these deleted files, you can simply re-pack or compress the disc and all the active files will be shifted down until there are no deleted files hanging around using up space.

FODS has this ability to repack a disc.

When saving and loading in FODS you work with named files, not track and sector data or I.D. bytes. This makes life alot easier. I've seen some disc systems where you have to specify track and sector info and/or I.D. bytes. What a pain that can be!

If you just want to save a source file temporarily, you can do that on what's known as "scratch-pads". There are two of these on a disc, "scratch-pade A" and "scratch-pad B", each of these temporary disc files can hold up to 16K or if "B" is not used, "A" can hold one file up to 32K in length. The only files that can be temporarily saved on scratch pad are files that have been built using the system text editor.

Being a dyed in the wool assembly language programmer, I really appreciate the FODS text editor! This line oriented editor is upwards compatible with the MOS/ARESCO editor but includes about everything you could aske for in a line editor. There is a full and semi-automatic line numbering feature, lines can be edited while they are being entered or recalled and edited later, strings can be located and substituted, the line numbers can be resequenced, the file size can be found, the hex address of a line can be known and comments can be appended to an assembly file after it has been found correct. Oops! I forgot to say lines can also be moved around and deleted. This isn't the complete list of FODS editor commands, just the ones that immediately come to mind.

Another very powerful feature of the system is the ability to actually execute a file containing a string of commands. For example, the newsletter mailing list is now being stored on disc. When I want to make labels, I would normally have to load each letter file and run the labels printing program. But with FODS, I can build up a "JOB" file of commands and execute it.

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

ANNOUNCING:

COMPUTER PROGRAMS FOR THE KIM-1

HDE ASSEMBLER

An advanced, two pass assembler using 6502 cross-assembler mnemonics. Free form, line oriented entry. Directives include: .OPTION, .BYTE, .WORD, .FILE, .OFFSET, .END. Output options include LIST, NOLIST, SYMBOL TABLE, NO SYMBOL TABLE, GENERATE, NOGENERATE, ERRORS, NOERRORS, TAB, NOTAB. Assemble using single or multiple files. Place source, object and symbol table anywhere in memory. Automatic paging, with header and page number. Approximately 4K. Loads at E000 or 2000.

HDE TEXT OUTPUT PROCESSING SYSTEM (TOPS)

A comprehensive output processor, including left, right and full justification, variable page length, page numbering (Arabic, capital and lower case Roman), page titling, leading and trailing edge tabbing, field sequence modification, selective repeat, selective page output, etc. More than 30 commands to format and control output of letters, documents, manuscripts and textual material of all types. Approximately 4K. Loads at E100 or 2100.

HDE DYNAMIC DEBUGGING TOOL (DDT)

Built-in assembler/disassembler coupled with single step and dynamic breakpoint entry/deletion facilitates rapid isolation, correlation and test of programs. Keystrokes minimized with single letter, unshifted commands, and optional arguments. Approximately 2K. Loads at E000 or 2000.

WATCH FOR PRODUCT ANNOUNCEMENTS THIS YEAR TO INCLUDE:

- Dual Channel RS-232 interface with auto answer
- Eprom card, eprom programmer
- NEC, Diablo printer interface.

All HDE memory and interface cards are KIM-4 compatible in a 4½" X 6½" format with on-board 5 volt regulation and address selection switches. Complex circuits are socketed for ease of maintenance. All products include a 90 day full parts and labor warranty, except memory boards which are 6 months, and computer programs which have a limited warranty.

ORDER DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER	PLAINSMAN MICROSYSTEMS	ARESCO
Box 523	Box 1712	P.O. Box 43
Medina, Ohio 44256	Auburn, Ala. 36830	Audubon, Pa. 19407
216-725-4560	800-633-8724	215-631-9052