6502 USER NOTES

no.13 **$2.50**

EDITORIAL

As you can tell already, we're back to using our old title. Although "USER NOTES: 6502 seemed like a good idea at first, old ties are hard to break - back to 6502 USER NOTES. It's easier to say anyway.

Lots of new things have been happening with the 6502 - many more are in store. The software situation has certainly gotten better - but there's still alot of room for improvement.

One problem that has slowed software development a bit is the fact that there have been no hobby mainframe systems (such as Southwest Techs 6800 machine and the IMSAI 8080 system) designed specifically for the 6502 to reach any level of popularity with aftermarket accessory manufacturers (which is a very good indication of marketplace acceptance).

By the way, I define "mainframe" as a backplane (motherboard) and a power supply in a box without an integral CPU.

Most 6502 hardware developers have gone their separate ways with regards to expansion capability. Witness the fact that there are now at least 6 bus oriented 6502 expansion systems which aren't the least bit compatible with each other.

Everybody loses in this situation. The hobbyist loses because since he will end up being locked into whatever system he purchases, he has to be sure that particular system has,(or will have) everything he has decided he needs (or will need). A very difficult decision to make for someone just getting into this hobby. One that could drive some folks away from the 6502 CPU altogether.

The manufacturer loses because with so many different 6502 expansion methods available, no self-respecting aftermarket supplier of boards would think of entering into such a diluted market. He would most likely go to the S-100 (IMSAI) or S-50 (SWTP) marketplace because of the numbers involved, the proliferation of software, and psuedo-standarization of hardware in those markets.

At this point, there is only one expansion bus which is being supported by aftermarket suppliers. That's the S-44 KIMbus from MOS Technology.

There are 6 companies (including MOS) supporting this bus in the form of accessory boards. That number is sure to increase since Synertek and Rockwell machines will also be using the S-44 KIMbus.

The S-44 KIMbus seems to hold the only real hope of popularizing the 6502 CPU and providing the consumer with an "intelligent" alternative to the S-100 bus and multiple sources of accessory boards.

When more than one company supports a particular bus in the form of accessory boards - everyone wins. The consumer now has the ability to shop around and look for the best deal on a particular board he has in mind. The supplier wins because as the market gets larger and broader - in its appeal, more consumers will enter into it and, as a result, more dollars will have a chance to reach him.

It will be interesting to watch how things develop in this marketplace.

6502 USER NOTES is published bimonthly (whenever possible) by Eric C. Rehnke, POB 33093, North Royalton, Ohio 44133. 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 freely reprinted by club newsletters as long as proper publication credit is given and the publisher is provided with a copy of the publication.

I think you're gonna like our new format alot. We've organized that articles to make things easier to find and are retypingall the articles (except for some program listings) to make things more consistent. Let me know your opinion. What would you like to see in our newsletter? I really enjoy feedback and look forward to YOUR comments.

They certainly are some neat new 6502 based machines entering the marketplace. Of course, I'm referring to the SYM (formaly VIM) from Synertek, the AIM from Rockwell and the Challenger 1P from OSI.

Phil Johnson (Johnson Computer) brought two OSI Challengers over to my place for a little demo so I could get an idea of what OSI was doing lately. I must say that I was impressed with the amount of capability built-in to these machines for the price. For example, for $350 you can get a machine with 8K Microsoft Basic on ROM, a 32 character/line video interface, built-in cassette interface, a metal box with built-in full size ASCII keyboard, character graphics capability, 4K RAM (expandable to 8K on board), a machine language monitor that lets you examine/change memory, and expansion capability (to OSI's bus, of course). Whether or not you can live with a 32 character display (24 character if you use an RF modulator) is up to you, but for all the obvious benefits of such a machine, that may not be a critical disadvantage.

About the only thing really missing on the Challenger 1P is a user I/O port and interval timer. These would have to be added to do any useful hacking. There is an expansion connector with the address, data and control busses but I don't know if the signals are buffered. I'll try to get more details on this for upcoming issues.

In all fairness to you, the reader, I feel it should be mentioned that I have talked to a number of people who had complaints about the level of service and support they received from OSI. If any of you have dealt with OSI lately, I'd be interested in hearing about your experiences.

The Synertek SYM certianly has some very interesting things to offer.

Its list of good points include on-board RAM, EPROM, and I/O expansion capability, a powerful monitor and a high-speed (1500 baud) cassette interface. Obviously, SYM's creators were working to update and improve on the basic KIM design.

I could tell by the number of on-board strapping options and software switching logic that this machine was meant to be as versatile as possible.

How the SYM "stacks-up" will be the subject of future articles.

Rockwells bid for marketplace superiority is called the AIM 65. This is actually a two board machine - on one board is a full size ASCII style keyboard while the other holds the rest of the system.

AIM is unique in that it contains a 20 column thermal printer besides a 20 column alphanumeric LED display. Like SYM, AIM has on-board EPROM and RAM expansion capability and an advanced monitor. Its on-board printer would make it a likely candidate for the process control and system monitoring environment.

SYM and AIM both have expansion connectors configured to fit the standard KIM-4 motherboard.

Articles on both these machines will be published in the next issue.

Hudson Digital Electronics (see back cover) has been making great advances in S-44 KIMbus compatible hardware and software products. The one thing I most admire about this firm is their way of introducing new products.

continued on page 28

software feature: KIM HEXPAWN

From Robert C. Leedom, 14069 Stevens Valley Ct., Glenwood, MD 21738

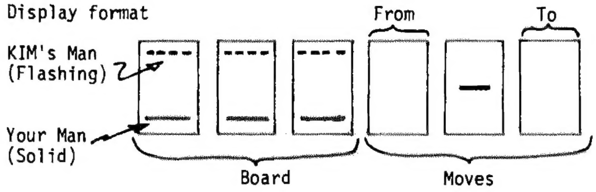
I was relieved to see (in Issue #12) that nobody's yet published a version HEXPAWN for KIM. I got my KIM in April, wrote HEXPAWN in May, and today (16 Oct 78) finally finished typing the listing. HEXPAWN first appeared in SCIENTIFIC American (Vol. 206, No. 3, Martin Gardner's "Mathematical Games"). The game is played on a 3 X 3 board. Each of the two players has three pieces, which move as chess pawns (move one square forward to vacant square, capture by moving one square diagonally to enemy piece's square). Object: get to your opponent's side of the board, or block him so that he cannot move.

This version was inspired by an article in the November 1975 BYTE, written by Bob Wier (with whom I corresponded on the subject of a "Super Star Trek" game in BASIC). Bob had written a HEXPAWN program for a 16-bit machine, and it took 4218 bytes (I assume they were 8-bit bytes). Unfortunately, (a) I have only the KIM-1 memory (and no access to an assembler), (b) the article only gave a general (top-level) flowchart and a move table, and (c) the article "Table of all Possible Board Positions and Moves" was both incomplete and incorrect, a fact I discovered only when I tried to play the game against my version of the program. Eventually, I solved problems (a), (b), and (c); here's the result:

Features of HEXPAWN for KIM-1

(a) Board coordinate 0 1 2 KIM's Men at 0, 1, 2
3 4 5
6 7 8 Your Men at 6, 7, 8

(b) Display format



(c) Program checks for (and only accepts) legal moves.

(d) KIM selects moves randomly, but learns. When When the computer loses, KIM's losing move is removed from the move table. Therefore, eventually (after 30 or so games) KIM should have only winning moves to select from!

(e) Two startup locations provided:
- (1) Full initialization -- all possible KIM moves restored to move table. (Start at $100).
- (2) New game initialization -- sets up board to play next game, but retains knowledge of previous bad moves. (Start AT $200).

(f) To allow tabulation/examination of the "learning" sequence, press and hold DA (Data Analysis) key at any time to display move # (0, 1, or 2 - there are three possible moves stored for each board position), Board index (see table at $10F) and Game number. Resume play upon release.

(g) Press PC (Person Concedes) to concede game to KIM.

(h) After loading program, enter AD, 0100, GO. At any time, to restart the current game, press GO.

P.S. Have been using Radio Shack Supertape with a K-Mart (S.S. Kresge Co.) Model 6-33-01 cassette recorder (cost about $27) with 100% success using Hypertape program. However, in tape exchanges, others can only read my tapes about 75% of the time, and I have slightly less success reading theirs.

```
                 ; Page 0 locations used by program HEXPAWN
0000      FLSHR    RES  1     Timer for flashing KIM's men
0001      DBD      RES  3     Current board in Display format
0004      MASK     RES  3     Masks for flashing KIM's men
0007      EBD      RES  9     Current bd - Easy-to-read format
0010      WINDO    RES  6     Current 7-segment display
0017      MOVTYP   RES  1     KIM's last move (TO;FROM)
0018      TOG      RES  1     On/off indicator for KIM's men
0019      GAMNUM   RES  1     Game number
001A      BDNDX    RES  1     3*Bd # for model match/move select
001B      MOVNO    RES  1     KIM's last move # (0,1, or 2)
001C      PTO      RES  1     Person's last "to" move
001D      FROM     RES  1     If< 0, no "from" move yet;
                              if≥0, is equal to the "from" move
001E      TMP      RES  1
001F      TMP1     RES  1
0020      POINTER  RES  1
0021      POINTO   RES  1     Page # (ADH) of MOVES
0022      MPOINT   RES  1
0023      MPOIN1   RES  1     Page # (ADH) of messages
0024      BGEBD    RES  9     Beginning bd - Easy-format;
                                    03 = KIM
                                    00 = space
                                    01 = Person
002D      MOVES    RES 99     Table of possible moves is placed
                                 here by startup routine and
                                 is modified as KIM "learns."
```

```
                  ; HEXPAWN for KIM-1. © Copyright May 1978 R.C.Leedom
0100 A2 6E     HXPNST LDX #$6E        Transfer moves, beginning
0102 BD 7E 01  INLP   LDA SPOINO,X       board, and pointer ADH's
0105 95 21            STA POINTO,X       to page zero.
0107 CA               DEX
0108 10 F8            BPL INLP
010A 85 19            STA GAMNUM      Set game # to zero.
010C 4C 00 02         JMP INIT
```

```
                ; The following are the 33 board positions that
                ; the HEXPAWN program will recognize after the
                ; human opponent has moved.  The squares are num-
                ; bered according to the scheme shown in the
                ; comment field for CAPSET (location 03F8).
                ; Here, the pieces and spaces (K=KIM, P=person,
                ; and _=space) are packed by column -- that is,
                ; in groups: 0,3,6; 1,4,7; 2,5,8. (For segment-
                ; lighting, actual data is ordered 360,471,582.)
                ;                             Bd #   BDNDX
010F 43 0B 0B BDMDL KP_,K_P,K_P       0      0
0112 0B 0B 43       K_P,K_P,KP_       1      3
0115 0B 43 0B       K_P,KP_,K_P       2      6
0118 C3 40 0B       KK_,_P_,K_P       3      9
011B 40 C3 0B       _P_,KK_,K_P       4      C
011E 43 48 03       KP_,_PP,K__       5      F
0121 43 03 48       KP_,K__,_PP       6     12
0124 08 C3 43       __P,KK_,KP_       7     15
0127 C8 43 43       _KP,KP_,KP_       8     18
012A C3 08 43       KK_,__P,KP_       9     1B
012D 43 43 C8       KP_,KP_,_KP       A     1E
0130 48 03 43       _PP,K__,KP_       B     21
0133 40 40 43       _P_,_P_,KP_       C     24
0136 08 43 03       __P,KP_,K__       D     27
0139 43 00 0B       KP_,___,K_P       E     2A
013C C0 C0 43       _K_,_K_,KP_       F     2D
013F 43 40 40       KP_,_P_,_P_      10     30
0142 C0 43 40       _K_,KP_,_P_      11     33
0145 40 43 C0       _P_,KP_,_K_      12     36
0148 C3 C0 40       KK_,_K_,_P_      13     39
014B 43 08 C3       KP_,__P,KK_      14     3C
014E 00 43 0B       ___,KP_,K_P      15     3F
0151 40 C0 C3       _P_,_K_,KK_      16     42
0154 C0 40 03       _K_,_P_,K__      17     45
0157 40 C3 00       _P_,KK_,___      18     48
015A 00 C3 40       ___,KK_,_P_      19     4B
015D 0B 00 43       K_P,___,KP_      1A     4E
0160 00 40 C3       ___,_P_,KK_      1B     51
0163 0B 40 C3       K_P,_P_,KK_      1C     54
0166 03 48 43       K__,_PP,KP_      1D     57
0169 43 C8 43       KP_,_KP,KP_      1E     5A
016C 00 43 00       ___,KP_,___      1F     5D
016F C3 40 00       KK_,_P_,___      20     60
              ;
              ; End-game messages
0172 3E 00 38 KWIN    DATA $3E,00,$38,$3F,$6D,$79
0175 3F 6D 79
0178 00 54 1C PWIN    DATA 00,$54,$1C,$78,$6D,00
017B 78 6D 00
     72       KWAD    EQU  @KWIN-$100
     78       PWAD    EQU  @PWIN-$100
              ;
              ; The following data is saved here for startup
              ;                initialization.
017E 00       SPOINO $00
017F 00       SMPOIN $00
0180 01       SMPOI1 $01
```

```
0181 03 03 03 SBGEPD DATA 03,03,03
0184 00 00 00        DATA 00,00,00
0187 01 01 01        DATA 01,01,01
018A 31 41 52 SMVTBL DATA $31,$41,$52 BDNDX 0
018D 30 41 51        DATA $30,$41,$51       3
0190 30 40 00        DATA $30,$40,0         6
0193 40 42 63        DATA $40,$42,$63       9
0196 31 52 74        DATA $31,$52,$74       C
0199 40 42 52        DATA $40,$42,$52       F
019C 31 41 51        DATA $31,$41,$51      12
019F 51 64 74        DATA $51,$64,$74      15
01A2 51 42 00        DATA $51,$42,0        18
01A5 63 73 00        DATA $63,$73,0        1B
01A8 40 31 00        DATA $40,$31,0        1E
01AB 31 41 51        DATA $31,$41,$51      21
01AE 42 00 00        DATA $42,0,0          24
01B1 42 52 00        DATA $42,$52,0        27
01B4 52 00 00        DATA $52,0,0          2A
01B7 63 74 00        DATA $63,$74,0        2D
01BA 40 00 00        DATA $40,0,0          30
01BD 51 63 00        DATA $51,$63,0        33
01C0 31 85 00        DATA $31,$85,0        36
01C3 63 74 00        DATA $63,$74,0        39
01C6 75 85 00        DATA $75,$85,0        3C
01C9 42 52 00        DATA $42,$52,0        3F
01CC 74 85 00        DATA $74,$85,0        42
01CF 63 42 52        DATA $63,$42,$52      45
01D2 74 31 00        DATA $74,$31,0        48
01D5 74 51 00        DATA $74,$51,0        4B
01D8 30 00 00        DATA $30,0,0          4E
01DB 42 85 00        DATA $42,$85,0        51
01DE 30 40 85        DATA $30,$40,$85      54
01E1 30 40 42        DATA $30,$40,$42      57
01E4 00 00 00        DATA 0,0,0            5A
01E7 00 00 00        DATA 0,0,0            5D
01EA 63 40 00        DATA $63,$40,0        60
```

```
0200 A2 07     INIT    LDX #$07       Initialize right
0202 A9 00             LDA #$00          side
0204 95 10     INITLP  STA WINDO,X       of display
0206 CA                DEX               (plus MOVTIM, MOVTYP)
0207 10 FB             BPL INITLP        to await
0209 A9 C0             LDA #$C0          person's
020B 85 14             STA WINDO+4       move.
020D A2 08             LDX #$08
020F B5 24     BDINIT  LDA BGEBD,X    Transfer beginning board (in
0211 95 07             STA EBD,X         Easy-format) to current
0213 CA                DEX               board.
0214 10 F9             BPL BDINIT
0216 86 1D             STX FROM       Indicate no "from" move yet.
0218 A2 03     DISPLT  LDX #$03       Clear the
021A A9 00     DSPLP   LDA #$00          "Display-format"
021C 95 00             STA FLSHR,X       board and
021E CA                DEX               the flasher-timer.
021F 10 F9             BPL DSPLP
0221 A0 02             LDY #$02       Start with 3rd char of board.
0223 18                CLC
0224 84 1E     NXDIG   STY TEMP
0226 A9 06             LDA #$06       Set up X to start with
0228 65 1E             ADC TEMP          lower segment for
022A AA                TAX               this character.
022B A9 00             LDA #$00       Clear A so can OR segments.
022D 6A 6A     NXSEG   ROR ROR        In this loop, shift the
022F 6A                ROR               segments into place.
0230 15 07             ORA EBD,X      OR 3 for KIM, 1 for person.
0232 CA CA             DEX DEX        Point to next
0234 CA                DEX               higher segment.
0235 10 F6             BPL NXSEG      Loop till character done.
0237 99 01 00          STA DBD,Y      Save completed char; go
023A 88                DEY               do next one to
023B 10 E7             BPL NXDIG         the left.
```

```
                  ; Main loop begins here
023D C6 00        DISPLO DEC FLSHR      Time to flip KIM bits?
023F 10 2E               BPL LITEST     No, just show current pattern.
0241 A9 30               LDA #$30       Yes.  Reset
0243 85 00               STA FLSHR         timer.
0245 A2 02               LDX #$02       Form the
0247 B5 01        GETMSK LDA DBD,X         flasher-mask
0249 4A                  LSR               patterns
024A 29 49               AND #$49          for
024C 95 04               STA MASK,X        the
024E CA                  DEX               current
024F 10 F6               BPL GETMSK        board.
0251 A2 02               LDX #$02       Set X for next loop.
0253 A5 18               LDA TOG        Toggle to
0255 49 80               EOR #$80          alternate 1's and 0's
0257 85 18               STA TOG           for KIM's men.
0259 30 09               BMI WNDSET     Go do 0's.
025B A9 00               LDA #$00
025D 95 04        ZERMSK STA MASK,X     Clear masks so
025F CA                  DEX               can do
0260 10 FB               BPL ZERMSK        1's.
0262 A2 02               LDX #$02
0264 B5 01        WNDSET LDA DBD,X      Use the
0266 29 49               AND #$49          masks
0268 55 04               EOR MASK,X        to flip
026A 95 10               STA WINDO,X       the bits.
026C CA                  DEX
026D 10 F5               BPL WNDSET
                  ; Output to KIM's 7-segment displays
026F A9 7F        LITEST LDA #$7F       Set directional
0271 8D 41 17            STA PADD          registers.
0274 A0 00               LDY #$00
0276 A2 09               LDX #$09       Start with leftmost char.
0278 B9 10 00 LITE       LDA WINDO,Y    Get character.
027B 84 FC               STY TEMP
027D 20 4E 1F            JSR CONVD+6    Output character.
0280 C8                  INY
0281 C0 06               CPY #$06       Done all six yet?
0283 90 F3               BCC LITE       Not yet, continue.
0285 20 3D 1F            JSR $1F3D      Turn off digits.
                  ; Keyboard input begins here
0288 D8           KEYGET CLD
0289 20 40 1F            JSR KEYIN
028C 20 6A 1F            JSR GETKEY
028F C9 13               CMP #$13       GO key?
0291 D0 03               BNE DACHK
0293 4C 00 02            JMP INIT       Yes, start new game.
0296 C9 11        DACHK  CMP #$11       DA key?
0298 D0 0E               BNE GIPROG
029A A2 02               LDX #$02       Yes, display (for Data Analysis)
029C B5 19        DALP   LDA GAMNUM,X      from left to right:
029E 95 F9               STA INH,X         Move # (00,01, or 02),
02A0 CA                  DEX               Board index (Bd # * 3),
02A1 10 F9               BPL DALP          Game #. (2 digits each)
02A3 20 1F 1F            JSR SCANDS        Keep doing this till DA
02A6 10 E0               BPL KEYGET        released; then resume play.
02A8 A6 14        GIPROG LDX WINDO+4    Is game still
02AA E0 C0               CPX #$C0          in progress?
02AC D0 C1               BNE LITEST     No. Keep showing endgame msg.
02AE C9 14               CMP #$14       PC key?
02B0 F0 78               BEQ KWLINK     Yes, Person Concedes.
02B2 A6 16               LDX MOVTIM     Person's turn to move?
02B4 D0 67               BNE TIMEDS     No, go time display.
02B6 20 C0 03            JSR LEGMOV     Yes. Did he make legal move?
02B9 10 35               BPL PERLM      Yes. Go execute it.
02BB A9 08               LDA #$08       He didn't make a legal move,
02BD 85 1E               STA TMP           does he have one?  Try
02BF 20 C0 03 LMCHK      JSR LEGMOV        each position to see.
02C2 10 0D               BPL TOMVCK

02C4 C6 1E        NXFMCK DEC TMP        Try
02C6 A5 1E               LDA TMP           next
02C8 10 F5               BPL LMCHK         position.
02CA 85 16               STA MOVTIM     Tried all, no luck, no legal
02CC 85 17               STA MOVTYP        moves possible.  Set KIM
02CE 4C 3D 02 FNMVLP     JMP DISPLO        win display after delay.
                  ; Continue looking for valid move for person
02D1 A5 1D        TOMVCK LDA FROM       Was valid move a "to" move?
02D3 10 F9               BPL FNMVLP     Yes, he can therefore move.
02D5 86 1D               STX FROM       No. Given this "from" move,
02D7 A9 06               LDA #$06          try all possible
02D9 85 1F               STA TMP1          "to" moves.
02DB 20 C0 03 LTMCHK     JSR LEGMOV     Find one?
02DE 10 0A               BPL OKMOV         Yes. He's got a move.
02E0 C6 1F               DEC TMP1       Try
02E2 A5 1F               LDA TMP1          next
02E4 10 F5               BPL LTMCHK        position.
02E6 86 1D               STX FROM       Tried all "to" moves; look
02E8 30 DA               BMI NXFMCK        for another "from" move.
                  ; Have found a possible "from-to" move for person.
02EA A9 FF        OKMOV  LDA #$FF       Has got a move he could make,
02EC 85 1D               STA FROM          so restore FROM and
02EE 30 DE               BMI FNMVLP        continue the game.
```

```
              ; Person has entered a legal move.
02F0 A9 00    PERLM  LDA #$00       Clear the "to" indication
02F2 85 15           STA WINDO+5       left from KIM's move.
02F4 A5 1D           LDA FROM       Was this a "from" move?
02F6 30 1B           BMI FRMDIS     Yes, display it; save move.
02F8 86 1C           STX PTO        No, save as person's "to".
02FA A0 01           LDY #$01       Set "person" indicator.
02FC A9 FF    TMINIT LDA #$FF       Set timer: not person's move.
02FE 85 16    MAKMOV STA MOVTIM     Save move-timer.
0300 94 07           STY EBD,X      Place piece on board.
0302 BD E7 1F        LDA DIGCOD,X   Get 7-segment code for
0305 85 15           STA WINDO+5       "to" indication on board.
0307 A0 00           LDY #$00       Remove piece from
0309 A6 1D           LDX FROM          previous
030B 94 07           STY EBD,X         board position.
030D A9 FF           LDA #$FF       Prepare for next
030F 85 1D           STA FROM          "from" move.
0311 30 02           BMI DISX       Go show this "from" move.
              ; This code displays "from" moves.
0313 86 1D    FRMDIS STX FROM       Save "from" move.
0315 BD E7 1F DISX   LDA DIGCOD,X   Use "from" in X to get 7-seg
0318 85 13           STA WINDO+3       indication.
031A 4C 18 02        JMP DISPLT     Return to main loop.
              ;
031D A5 18    TIMEDS LDA TOG        Time to decrement move timer?
031F 30 AD           BMI FNMVLP     Not yet.
0321 C6 16           DEC MOVTIM     Yes. Ready for next move?
0323 D0 A9           BNE FNMVLP     Not yet.

0325 A5 17    KWCHK  LDA MOVTYP     Has KIM moved
0327 4A              LSR               to either
0328 C9 30           CMP #$30          6,7,or 8?
032A 10 5B    KWLINK BPL KIMWIN     Yes. KIM won.
032C A5 1C           LDA PTO        Has person moved
032E C9 03           CMP #$03          to 0,1,or 2?
0330 30 44           BMI PERWIN     Yes.  Person won.
              ; Try to match current board with stored model.
0332 A0 60           LDY #$60       (#models - 1)*3 = 32*3 = 96
0334 BE 0F 01 MDLCHK LDX BDMDL,Y
0337 E4 01           CPX DBD        First column match?
0339 D0 0E           BNE NXBD       No, try next board model.
033B BE 10 01        LDX BDMDL+1,Y  Yes, does
033E E4 02           CPX DBD+1         second column match?
0340 D0 07           BNE NXBD       No, try next board model.
0342 BE 11 01        LDX BDMDL+2,Y  Yes, does
0345 E4 03           CPX DBD+2         third column match?
0347 F0 07           BEQ GOTMDL     Yes, found model. Go get move.
0349 88 88    NXBD   DEY DEY        Point to
034B 88              DEY               next board model
034C 10 E6           BPL MDLCHK        and keep comparing.
034E 30 33           BMI PWMSG      No models found; have KIM
                                       concede the game.
              ; Pick one of the remaining moves for this position.
0350 AD 04 17 GOTMDL LDA TIMER      Use the timer to
0353 29 03           AND #$03          arbitrarily select
0355 AA              TAX               move 0, 1, or 2.
0356 F0 01           BEQ POK           (This code picks #2
0358 CA              DEX               half the time.)
0359 A9 02    POK    LDA #$02       Inititialize the counter for
035B 85 1E           STA TMP           how many moves to try (3).
035D 86 1F    MVSLLP STX TMP1       Temporary move number.
035F A9 2D    MVLP1  LDA ADMVTB
0361 18              CLC
0362 65 1F           ADC TMP1       Set ADL of pointer to pick
0364 85 20           STA POINTER       up this move.
0366 B1 20           LDA (POINTER),Y
0368 D0 30           BNE GOTMOV     Got a valid move -- use it!
036A C6 1E           DEC TMP
036C 30 08           BMI PERWIN     No moves left; KIM resigns.
036E C6 1F           DEC TMP1       Try next move
0370 10 ED           BPL MVLP1         in the set. (May try
0372 A2 02           LDX #$02          in order 2,1,0; 1,0,2;
0374 10 E7           BPL MVSLLP        or 0,2,1.)
0376 A4 1A    PERWIN LDY BDNDX      Person has won.   ************
0378 18              CLC            Compute
0379 A9 2D           LDA ADMVTB        the ADL
037B 65 1B           ADC MOVNO         of KIM's
037D 85 20           STA POINTER       last move.(POINTER+1 = 0)
037F A9 00           LDA #$00       Wipe out the last
0381 91 20           STA (POINTER),Y   move KIM made.
0383 A9 78           LDA PWAD       Get address of "person won"
0385 10 02           BPL STAD          message.
0387 A9 72    KIMWIN LDA KWAD       Get "KIM won" msg address.
0389 85 22    STAD   STA MPOINT     Point to message address (ADL)

              ; Display end-of-game message
038B A0 05           LDY #$05
038D B1 22    FILWIN LDA (MPOINT),Y Store the six-letter
038F 99 10 00        STA WINDO,Y       message in the window.
0392 88              DEY
0393 10 F8           BPL FILWIN
0395 E6 19           INC GAMNUM     Increment game number.
0397 4C 6F 02        JMP LITEST     Show msg and wait for GO.
```

```
                     ; Make KIM's chosen move
039A 85 17      GOTMOV STA MOVTYP    Save move for later checks.
039C 84 1A             STY BDNDX     Save board pointer.
039E A6 1F             LDX TMP1      Pick up
03A0 86 1B             STX MOVNO        move # (=0,1, or 2).
03A2 48 48             PHA PHA       Save 2 copies of move type.
03A4 4A 4A             LSR LSR
03A6 4A 4A             LSR LSR
03A8 AA                TAX           Place "to" move in X.
03A9 68                PLA
03AA 29 0F             AND #$0F      Extract and save
03AC 85 1D             STA FROM         "from" move.
03AE A0 03             LDY #$03      Indicate KIM move being made.
03B0 68 4A             PLA LSR       Is upper half-byte of
03B2 C9 30             CMP #$30         move a 6,7, or 8?
03B4 30 03             BMI NOKWIN    No. KIM hasn't won.
03B6 4C FC 02          JMP TIMINIT   Yes. Show winning move.
03B9 A9 00      NOKWIN LDA #$00      Indicate it's person's
03BB 4C FE 02          JMP MAKMOV       move, and make KIM's.
                ; Subroutine to test for legal player move.
                ; Call: with move in A.
                ; Returns: with X = $FF if illegal move
                ;               X = move if legal
03BE 00 00
03C0 C9 09      LEGMOV CMP #$09      Is move 0 to 8?
03C2 B0 22             BCS MOVNFG    No, illegal.
03C4 AA                TAX
03C5 B5 07             LDA EBD,X     Extract player indicator from
03C7 6A 6A             ROR ROR          board: 1 player, 3 KIM.
03C9 10 27             BPL TOCHK     Nobody here, but OK if "to".
03CB A5 1D             LDA FROM      Is this a "from" move?
03CD 10 04             BPL PTOMOV    No. Go see if legal "to".
03CF B0 15             BCS MOVNFG    Yes, but KIM's here! Bad.
03D1 8A         MOVOK  TXA           Return. Legal move was made
03D2 60                RTS              (or found possible).
03D3 8A         PTOMOV TXA           Place "to" move in A.
03D4 90 10             BCC MOVNFG    Person here! Can't capture!
03D6 0A 0A             ASL ASL       KIM here. A capture. Can
03D8 0A 0A             ASL ASL          only be one of 8 possible
03DA 05 1D             ORA FROM         moves. Format "TO:FROM",
03DC A0 07             LDY #$07         and test against
03DE D9 F8 03   CAPCHK CMP CAPSET,Y     each possibility.
03E1 F0 EE             BEQ MOVOK     Found it! Move is OK.
03E3 88                DEY
03E4 10 F8             BPL CAPCHK
03E6 A2 FF      MOVNFG LDX #$FF      Move illegal. Set indicator.
03E8 60                RTS           (Second of two return points.)

03E9 8A         NOCAP  TXA           If here, not capture, C=0.
03EA 69 03             ADC #$03      Is move + 3 = FROM?
03EC C5 1D             CMP FROM
03EE D0 F6             BNE MOVNFG    No.  Illegal move.
03F0 F0 DF             BEQ MOVOK
03F2 A4 1D      TOCHK  LDY FROM      Space here. "To" move?
03F4 30 F0             BMI MOVNFG    No. Illegal move.
03F6 10 F1             BPL NOCAP     Yes. See if valid move.
                ;
                ; Set of all possible "capture" moves by person,
                ;   packed in "TO:FROM" format.
                ;
                ;   Captures are from to
03F8 13         CAPSET DATA $13        3   1
03F9 04                DATA $04        4   0
03FA 24                DATA $24        4   2
03FB 15                DATA $15        5   1
03FC 46                DATA $46        6   4
03FD 37                DATA $37        7   3
03FE 57                DATA $57        7   5
03FF 48                DATA $48        8   4
                ;
```

Board format

0	1	2
3	4	5
6	7	8

HEXPAWN HEX DUMP

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100 A2 6E BD 7E 01 95 21 CA 10 F8 85 19 4C 00 02 43
0110 0B 0B 03 0B 43 0B 43 0B C3 40 0B 40 C3 0B 43 48
0120 03 43 03 48 08 C3 43 C8 43 43 C3 08 43 43 43 C8
0130 48 03 43 40 40 43 08 43 03 43 00 0B C0 C0 43 43
0140 40 40 C0 43 40 40 43 C0 C3 C0 40 43 08 C3 00 43
0150 0B 40 C0 C3 C0 40 03 40 C3 00 00 C3 40 0B 00 43
0160 00 40 C3 0B 40 C3 03 48 43 43 C8 43 00 43 00 C3
0170 40 00 3E 00 38 3F 6D 79 00 54 1C 78 6D 00 00 00
0180 01 03 03 03 00 00 00 01 01 01 31 41 52 30 41 51
0190 30 40 00 40 42 63 31 52 74 40 42 52 31 41 51 51
01A0 64 74 51 42 00 63 73 00 40 31 00 31 41 51 42 00
01B0 00 42 52 00 52 00 00 63 74 00 40 00 00 51 63 00
01C0 31 85 00 63 74 00 75 85 00 42 52 00 74 85 00 63
01D0 42 52 74 31 00 74 51 00 30 00 00 42 85 00 30 40
01E0 85 30 40 42 00 00 00 00 00 00 63 40 00
```

```
0300 94 07 BD E7 1F 85 15 A0 00 A6 1D 94 07 A9 FF 85
0310 1D 30 02 86 1D BD E7 1F 85 13 4C 18 02 A5 18 30
0320 AD C6 16 D0 A9 A5 17 4A C9 30 10 5B A5 1C C9 03
0330 30 44 A0 60 BE 0F 01 E4 01 D0 0E BE 10 01 E4 02
0340 D0 07 BE 11 01 E4 03 F0 07 88 88 88 10 E6 30 33
0350 AD 04 17 29 03 AA F0 01 CA A9 02 85 1E 86 1F A9
0360 2D 18 65 1F 85 20 B1 20 D0 30 C6 1E 30 08 C6 1F
0370 10 ED A2 02 10 E7 A4 1A 18 A9 2D 65 1B 85 20 A9
0380 00 91 20 A9 78 10 02 A9 72 85 22 A0 05 B1 22 99
0390 10 00 88 10 F8 E6 19 4C 6F 02 85 17 84 1A A6 1F
03A0 86 1B 48 48 4A 4A 4A 4A AA 68 29 0F 85 1D A0 03
03B0 68 4A C9 30 30 03 4C FC 02 A9 00 4C FE 02 00 00
03C0 C9 09 B0 22 AA B5 07 6A 6A 10 27 A5 1D 10 04 B0
03D0 15 8A 60 8A 90 10 0A 0A 0A 0A 05 1D A0 07 D9 F8
03E0 03 F0 EE 88 10 F8 A2 FF 60 8A 69 03 C5 1D D0 F6
03F0 F0 DF A4 1D 30 F0 10 F1 13 04 24 15 46 37 57 48
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
```

```
0200 A2 07 A9 00 95 10 CA 10 FB A9 C0 85 14 A2 08 B5
0210 24 95 07 CA 10 F9 86 1D A2 03 A9 00 95 00 CA 10
0220 F9 A0 02 18 84 1E A9 06 65 1E AA A9 00 6A 6A 6A
0230 15 07 CA CA CA 10 F6 99 01 00 88 10 E7 C6 00 10
0240 2E A9 30 85 00 A2 02 B5 01 4A 29 49 95 04 CA 10
0250 F6 A2 02 A5 18 49 80 85 18 30 09 A9 00 95 04 CA
0260 10 FB A2 02 B5 01 29 49 55 04 95 10 CA 10 F5 A9
0270 7F 8D 41 17 A0 00 A2 09 B9 10 00 84 FC 20 4E 1F
0280 C8 C0 06 90 F3 20 3D 1F D8 20 40 1F 20 6A 1F C9
0290 13 D0 03 4C 00 02 C9 11 D0 0E A2 02 B5 19 95 F9
02A0 CA 10 F9 20 1F 1F 10 E0 A6 14 E0 C0 D0 C1 C9 14
02B0 F0 78 A6 16 D0 67 20 C0 03 10 35 A9 08 85 1E 20
02C0 C0 03 10 0D C6 1E A5 1E 10 F5 85 16 85 17 4C 3D
02D0 02 A5 1D 10 F9 86 1D A9 06 85 1F 20 C0 03 10 0A
02E0 C6 1F A5 1F 10 F5 86 1D 30 DA A9 FF 85 1D 30 DE
02F0 A9 00 85 15 A5 1D 30 1B 86 1C A0 01 A9 FF 85 16
```

6502 OP CODES

ARRANGED IN LOGICAL ORDER BY JIM BUTTERFIELD, TORONTO

	IMM 2	ZPAG 2	Z,X 2	Z,Y 2	ABS 3	A,X 3	A,Y 3
ASL		06	16		0E	1E	
ROL		26	36		2E	3E	
LSR		46	56		4E	5E	
ROR		66	76		6E	7E	
STX		86		96	8E		
LDX	A2	A6		B6	AE		BE
DEC		C6	D6		CE	DE	
INC		E6	F6		EE	FE	

OP CODE ENDS IN -2, -6, OR -E

	IMM 2	ZPAG 2	Z,X 2	ABS 3	A,X 3
BIT		24		2C	
STY		84	94	8C	
LDY	A0	A4	B4	AC	BC
CPY	C0	C4		CC	
CPX	E0	E4		EC	

MISC. -0, -4, -C

	IMM 2	ZPAG 2	Z,X 2	(I,X) 2	(I),Y 2	ABS 3	A,X 3	A,Y 3
ORA	09	05	15	01	11	0D	1D	19
AND	29	25	35	21	31	2D	3D	39
EOR	49	45	55	41	51	4D	5D	59
ADC	69	65	75	61	71	6D	7D	79
STA		85	95	81	91	8D	9D	99
LDA	A9	A5	B5	A1	B1	AD	BD	B9
CMP	C9	C5	D5	C1	D1	CD	DD	D9
SBC	E9	E5	F5	E1	F1	ED	FD	F9

OP CODE ENDS IN -1, -5, -9, OR -D

BPL	10	BMI	30
BVC	50	BVS	70
BCC	90	BCS	B0
BNE	D0	BEQ	F0

BRANCHES -0

	ABS	(IND)
JSR	20	
JMP	4C	6C

JUMPS

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	BRK				RTI		RTS									
-8	PHP	CLC	PLP	SEC	PHA	CLI	PLA	SEI	DEY	TYA	TAY	CLV	INY	CLD	INX	SED
-A	ASL-A		ROL-A		LSR-A		ROR-A		TXA	TXS	TAX	TSX	DEX		NOP	

SINGLE-BYTE OP CODES -0, -8, -A

Another OP-CODE chart? Yes, but there is a reason.

This chart groups the codes logically. This way, you get three benefits.

First, you get to see how the codes are classified and decoded. A glance at the chart shows that LDA and ADC, for example, are close cousins: same addressing modes, same timing, and quite similar OP-CODES; on the other hand, LDA and LDX are noticeably different. The classification idea can be useful to those who want to dig into op-codes, say to write an assembler or a disassembler.

Secondly, it's handy for looking up an OP-CODE-maybe easier than an alphabetical list. You'll very quickly learn to look at the right box and spot the code you want right away. As you get used to the groupings, you'll also develop a feel for the addressing modes that are allowed.

Thirdly, you'll find it convenient for identifying an unknown op-code---("What the heck is CE, anyway?")

Jim B.

EDITORS NOTE: I have found this chart to be extremely useful in designing opcode decode algorithms etc.

cassette interface stuff:

TAPE VERIFY (II)

Dr. Barry Tepperman
25 St. Mary St., #411
Toronto, Ontario M4Y 1R2
Canada

The only major disadvantage apparent in James Van Ornum's "Tape Verify" routine (from "First Book of KIM") is that, located as it is in the KIM monitor's "volatile execution block" of RAM, it must be manually loaded for each use rather than loaded from tape or relocated for use in ROM. The following is a modification of this routine that treats "Tape Verify" as a block of data loaded as an array into VEB; it also appropriately zeros the checksums, so that (apart from loading this routine and starting it up - in this example at 0200) the only manual loading required is to make sure that the correct file ID is in 17F9. TAPE VERIFY II can be loaded from tape, or, being fully relocatable, be put into PROM for those of you (like me) whose expansion plans for KIM include an extended firmware operating system. As you might expect from description, data array PROG is a hex dump of the original TAPE VERIFY routine.

```
VERIFY   CLD             0200   D8
         LDA   $#00        01   A9   00
         STA   CHKL        03   8D   E7   17
         STA   CHKH        06   8D   E8   17
         LDX   $#0C        09   A2   0C
LOADP    LDA   PROG,X      0B   BD   17   02
         STA   VEB,X       0E   9D   EB   17
         DEX               11   CA
         BNE   LOADP       12   D0   F7
         JMP   $#188C      14   4C   8C   18
         BRK               17   00
```

data array: PROG

```
0218 CD 00 00 D0 03 4C 0F 19 4C 29 19 00
```

comapre with original TAPE VERIFY:

```
VEB      CMP   START      17EC   CD   00   00
         BNE   FAILED       EF   D0   03
         JMP   LOADT12      F1   4C   0F   19
FAILED   JMP   LOADT9       F4   4C   29   19
```

RADIO TAPE FEEDBACK

Daniel Gardner
11825 Beach Blvd.
Stanton, Ca 90680

Here is an interesting way to verify that KIM has found your program when loading from audio tape. All you need is an A.M. transistor radio (a KIM and a cassette recorder would be helpful too). Place the radio somewhere close to KIM and tune to a frequency where you can hear the "whine" of KIM's displays. Now, if you have already loaded the I.D. of your program, you are ready to verify a load. Enter AD 1 8 7 3, GO, and start your tape. You should now hear a buzz coming from the radio (you might have to fine tune it until you hear the buzz), mixed with the buzz are "clicks" as the microprocessor reads the synch. bytes. After awhile (100 synch bytes to be exact), if KIM ahs found your program the clicks will become more distinct, but if KIM didn't find your program the clicks will disappear. If you have ever waited 2 minutes for KIM to load a long program and found "she" didn't see it at all you'll appreciate this little trick. Thanks go to Scott Ogata for this idea.

RELIABILITY HINT

John Watney
24133 Young Court
Los Altos, Ca 94022

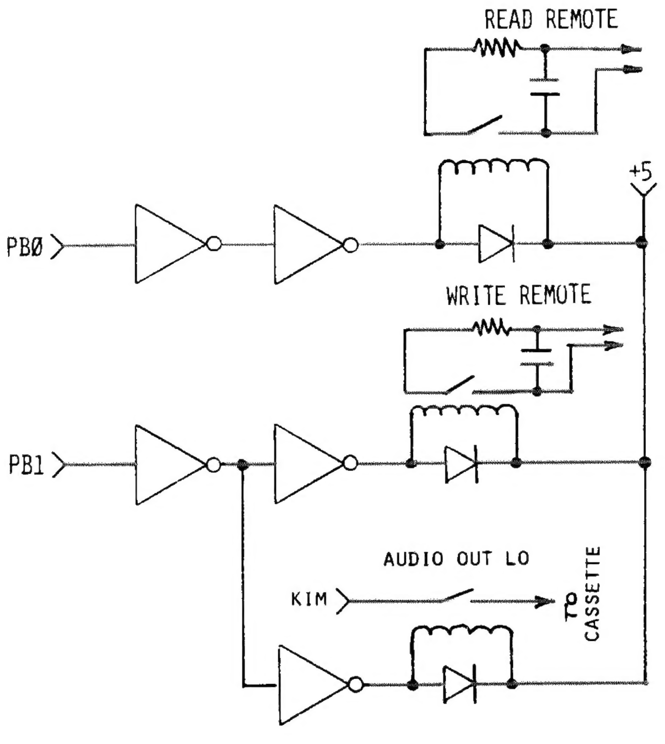
I have a hint that might be of interest to your readers. My cassette recorder gave unreliable results (on KIM) which were traced to low frequency noise, 60Hz and the like. Reliability was greatly improved by cutting the low frequency response with a 100 ohm load on the audio input coupling capacitor C6. It was conveniently soldered to the board between the junction of C6 and R8 and VCC at the junction of R14 and R15. In my system the attenuation of the 2.4 KHZ component of the play back signal brought it to the same level as the 3.7 KHZ component.

HELP relay package fixit

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

If you purchased the HELP Relay package from THE COMPUTERIST, you should know that the version of the circuit which has three relays will probably not work as shown in the wiring diagram. (An early version used two, until it was determined that a signal exists on the output to the recorder, which has to be interrupted.)

The diagram supplied with the set of parts shows a 7404 driving two relays. The relays I received draw about 14ma each, while the 7404 has a maximum rating of 16ma. The solution is to get another 1N914 for the third relay and follow the changed wiring below, which simply uses another buffer in the 7404. Other solutions using other chips are also possible. I have made the change (after burning out a 7404) and my unit now works.



TAPE FILE RECOVERY ROUTINE

Joel Swank
4655 SW 142nd #186
Beaverton, Or 97005

Ever have a tape file with a dropout? One that fails on the same byte every time. There must be good data behind that dropout, but how to get at it?

The normal tape read routine quits when it gets an invalid character. Instead the recovery routine flags the error by storing an asterisk(*) in memory, and begins reading bits looking for a valid character. When it gets one it resumes reading the file. The only problem is that there is no way of telling whether the first valid character is the first half of a byte or the last half of a byte. To overcome this problem the routine uses an external flag byte (HALF) to determine what to do with the first valid character after a dropout. A bit is shifted out of the high order end of HALF each time a recovery is attempted. If the bit is zero the first valid character is ignored. If it is one the first valid character is used to form the first valid byte. Upon each entry into recovery mode the counter ERRC is incremented. If it wraps to zero the program is aborted.

To recover a file initialize $17F5-$17F9 as usual and set HALF ($C8) to $00. Start the program at $200. When $FFFF appears look at ERRC ($C7). If the count is low then examine the data to find the errors marked '*' ($2A). Determine where the data is a half byte out of sync and set one bits in HALF accordingly. Rerun the program and the data, minus the dropouts should be in memory. I have recovered files with two dropouts, it should work for as many as 8.

If you have a file that has a dropout in the sync pattern and won't sync-up it may be recovered by using SCAN ($298) as entry point, effectively starting in recovery mode. You must first initialize VEBB ($17EC) with $8Dnnnn60, where nnnn is the address where the data is to be stored. Also zero ERRC. The ID and start address will be read and stored like data. It is also possible to begin reading files in the middle in this manner. The routine also performs the special tape read functions (ID=00 or FF). Thanks to Jim Butterfield for use of his synchronizaiton code.

```
1090                      :  ZERO PAGE STORAGE
1100                      :
1110                      HALF .DL ØØC8
1120                      ERRC .DL ØØC7     ERROR COUNT
1130                      INH  .DL ØØF9
1140                      :
1150                      :  EXTERNAL LABELS
1160                      :
1170                      VEBB .DL 17EC
1180                      SAL  .DL 17F5
1190                      SAH  .DL 17F6
1200                      EAL  .DL 17F7
1210                      EAH  .DL 17F8
1220                      ID   .DL 17F9
1230                      INTV .DL 1932
1240                      ENDØ .DL 1925
1250                      ENDF .DL 1929
1260                      INVB .DL 1932
1270                      RDBY .DL 19F3
1280                      RDCH .DL 1A24
1290                      CHKT .DL 194C
1300                      INC1 .DL 19EA
1310                      SED  .DL 1742
1320                      PAKT .DL 1AØØ
1330                      RDBT .DL 1A41
1340                      CHKL .DL 17E7
1350                      CHKH .DL 17E8
1360                      :
1370                      :  ENTRY POINT
1380                      :
1390 Ø2ØØ A9 8D           RECV LDA 8D       OPCODE FOR STA
1400 Ø2Ø2 80 EC 17             STA VEBB     INTO VEBB
1410 Ø2Ø5 A9 ØØ                LDA ØØ
1420 Ø2Ø7 85 C7                STA *ERRC    INIT COUNT
1430 Ø2Ø9 C9 FF                CMP ØFF
1440 Ø2ØB 2Ø 32 19             JSR INVB     INIT VEBB
1450 Ø2ØE A9 Ø7                LDA Ø7       CIRECTIONAL REG
1460 Ø21Ø 80 42 17             STA SBD
1470 Ø213 2Ø 41 1A        SYN  JSR RDBT     CET A BIT
1480 Ø216 46 F9                LSR *INH
1490 Ø218 Ø5 F9                ORA *INH     SHIFT INTO LEFT OF INH
1500 Ø21A 85 F9                STA *INH
1510 Ø21C C9 16           TST  CMP 16       SYNC CHARACTER?
1520 Ø21E DØ F3                BNE SYN      NO - KEEP LOOKIN
1530 Ø22Ø 2Ø 24 1A             JSR RDCH     GET A CHARACTER
1540 Ø223 C6 F9                DEC *INH     COUNT 22 SYNCS
1550 Ø225 1Ø F5                BPL TST
1560 Ø227 C9 2A                CMP '*       * FLAGS START OF RECORD
1570 Ø229 DØ F1                BNE TST      IF NOT - THEN MUST BE SYNC
1580 Ø22B 2Ø F3 19             JSR RDBY     CET BYTE
1590 Ø22E CD F9 17             CMP ID       CORRECT RECORD?
1600 Ø231 FØ 18                BEQ LOAD     YES - READ IT
1610 Ø233 AD F9 17             LDA ID       ID=Ø?
1620 Ø236 FØ 13                BEQ LOAD     YES - READ IT ANYWAY
1630 Ø238 C9 FF                CMP ØFF      ID=FF?
1640 Ø23A DØ D7                BNE SYN      NO TRY NEXT
1650 Ø23C 2Ø F3 19             JSR RDBY     YES IGNORE SA OF TAPE
1660 Ø23F 2Ø 4C 19             JSR CHKT
1670 Ø242 2Ø F3 19             JSR RDBY
1680 Ø245 2Ø 4C 19             JSR CHKT
1690 Ø248 38                   SEC
1700 Ø249 BØ 12                BCS BYTE     RELATIVE JUMP
1710 Ø24B 2Ø F3 19        LOAD JSR RDBY     READ START ADDRESS AND SAVE
1720 Ø24E 8D ED 17             STA VEBB+Ø1
1730 Ø251 2Ø 4C 19             JSR CHKT
1740 Ø254 2Ø F3 19             JSR RDBY
1750 Ø257 8D EE 17             STA VEBB+Ø2
```

```
1760 Ø25A 2Ø 4C 19           JSR CHKT
1770 Ø25D A2 Ø2        BYTE LDX Ø2        INDEX TO READ 2 CHAR BYTES
1780 Ø25F 2Ø 24 1A     CHAR JSR RDCH      GET A CHARACTER
1790 Ø262 C9 2F             CMP '/        / FLAGS END OF DATA
1800 Ø264 FØ 13             BEQ CHEK
1810 Ø266 2Ø ØØ 1A          JSR PAKT      CONVERT TO HEX NYBBLE
1820 Ø269 DØ 21             BNE BADC      INVALID CHARACTER
1830 Ø26B CA                DEX
1840 Ø26C DØ F1             BNE CHAR
1850 Ø26E 2Ø 4C 19          JSR CHKT      COMPUTE CHECKSUM
1860 Ø271 2Ø EC 17          JSR VEBD      STORE BYTE
1870 Ø274 2Ø EA 19          JSR INC↑      NEXT BYTE
1880 Ø277 DØ E4             BNE BYTE
1890                   ; GET CHECKSUM AND COMPARE TO COMPUTED VALUE
1900 Ø279 2Ø F3 19     CHEK JSR RDBY
1910 Ø27C CD E7 17          CMP CHKL
1920 Ø27F DØ 2D             BNE BADS
1930 Ø281 2Ø F3 19          JSR RDBY
1940 Ø284 CD E8 17          CMP CHKH
1950 Ø287 DØ 25             BNE BADS
1960 Ø289 4C 25 19          JMP ENDØ      NORMAL EXIT
1970                   ; ATTEMPT RECOVERY AFTER ERROR
1980 Ø28C A9 2A        BADC LDA '*        FLAG BAD BYTE
1990 Ø28E 2Ø EC 17          JSR VEBD
2ØØØ Ø291 2Ø EA 19          JSR INC↑
2Ø1Ø Ø294 E6 C7             INC *ERRC     COUNT ERROR
2Ø2Ø Ø296 FØ 16             BEQ BADS      ERROR COUNT OVERFLOW
2Ø3Ø Ø298 2Ø 41 1A     SCAN JSR RDBT      GET A BIT
2Ø4Ø Ø29B 46 F9             LSR *INH
2Ø5Ø Ø29D Ø5 F9             ORA *INH      SHIFT IN
2Ø6Ø Ø29F 85 F9             STA *INH
2Ø7Ø Ø2A1 2Ø ØØ 1A          JSR PAKT      GOT A VALID CHARACTER YET?
2Ø8Ø Ø2A4 DØ F2             BNE SCAN      NO KEEP TRYIN
2Ø9Ø Ø2A6 Ø6 C8             ASL *HALF     TEST NEXT SKIP BIT
21ØØ Ø2A8 9Ø B3             BCC BYTE      IGNORE THIS CHARACTER
211Ø Ø2AA A2 Ø1             LDX Ø1        ELSE USE IT AS FIRST HALF
212Ø Ø2AC DØ B1             BNE CHAR      AND GO READ 2ND HALF
213Ø Ø2AE 4C 29 19     BADS JMP ENDF      SHOW ERROR
214Ø                   END$ .EN
```

hey!! KIM SOFTWARE ON CASSETTE

Here, by popular demand, are the first of a series of cassette software offerings by the NOTES. Many of you have asked that some of the longer programs which are published in our newsletter be made available on cassette so that your time could be spent doing things besides punching in programs.

All cassettes will be original recordings (not copies) dumped directly from memory using the standard KIM recording format.

Besides HEXPAWN (our software feature) we also have KIMATH which is a 2K math subroutine package which was to be released in ROM from MOS Technology a couple of years ago - and wasn't. The KIMATH manual, which includes a complete source listings of the $F800 version (same as the $2000 version 'cept for the addresses) is available for $15.00 from sources that I know of.

Johnson Computer	AB Computers
P.O. Box 523	Box 104
Medina, Ohio 44256	Perkasie, Pa 18944
216-725-4560	215-257-8195

The KIMATH manual is not included with the cassette and must be purchased separately from one of the above sources.

An errata sheet will be included with each cassette with some corrections for the manual. By the way, we have the ability to reassemble KIMATH anywhere in memory for $5.00 extra.

Each cassette will have a 30 second "SYNC" leader which can be used for aligning your head (no, the one on your cassette) or PLL. The heads of the machines which will be used to record your cassette have been aligned from a Recording studio cassette which was set up with an alignment "standard".

KIMATH	(specify $2000 or $F800 version)	$12.00
HEXPAWN	($0100-$03FF)....................	$5.00

U.S. funds only. Overseas customers please include $1.00 extra for postage.

LANGUAGE LAB:

focal

At this point in time, FOCAL is the most documented of the high level languages which run on our beloved 6502. Having a complete source listing is definitely invaluable.

This openness on the part of the implementor has made it so easy to fidget around with FOCALs internals and even fix a problem or two.

One of the things that did sort of annoy me was the almost 1 character delay encountered when typing in FOCAL program text from a hard-copy terminal. (I have the Aresco version).

As it turns out, thanks to the source listing, I found that FOCALs author did some elaborate arm waving to prevent KIM from echoing the character which has been input to the TTY port. No small feat, I might add, since KIM echoes the tty input in hardware (not software!).

(If you're wondering how - FOCAL makes the terminal think that the character getting echoed is a RUBOUT character - which the terminal ignores).

Anyhow, I don't quite know why FOCAL bothers to do this - the character ends up getting echoed in software anyway. (There is a function which does enable to echo to be shut off completely).

Make the following changes to FOCAL. This patch was found in the FOCAL User Manual ($12.00 from the 6502 Program Exchange) and was apparently an update for FCL-65E.

```
34AA   84  A5        OUT    STY  SAVYR
                            ; save "Y"
34AC   20  A0  1E           JSR  OUTCH
34AF   A4  A5               LDY  SAVYR
                            ;restore "Y"
34B1   18                   CLC
                            ; indicate success
34B2   60            RTS
                            ; return


34B3   E6  76         IN    INC  HASH
                            ;bump random seed
34B5   2C  40  17           BIT  SAD
                            ; test input port
34B8   30  F9               BMI  IN
                            ; loop 'til start bit
34BA   A5  6B               LDA  ECHFLG
                            ; get echo flag
34BC   D0  03               BNE  NOECH
                            ; branch for no echo
34BE   4C  5A  1E           JMP  GETCH
                            ; get character with
                              echo

34C1   AD  42  17    NOECH  LDA  SBD
                            ; get port status
34C4   29  FE               AND  #FE
                            ; turn off bit
34C6   8D  42  17           STA  SBD
34C9   20  5A  1E           JSR  GETCH
34CC   48                   PHA
                            ; save character
34CD   AD  42  17           LDA  SBD
                            ; get port status
34D0   09  01               ORA  #01
                            ; turn on bit
34D2   8D  42  17           STA  SBD
                            ; make echo a rubout
34D5   A9  00               LDA  #0
                            ; get a null character
34D6   20  A0  1E           JSR  OUTCH
                            ; echo it
34D9   68                   PLA
                            ; restore input char.
34DA   18                   CLC
                            ; indicate success
34DB   60                   RTS
                            ; return
```

```
28F2   EA EA EA   was 20 02 29
35B4   B3         was A5
```

Faster typists will really notice a difference.

A really neat feature of FOCAL is the fact that you can add specialized functions.

Function calls consist of four (or fewer) letters beginning with the letter "F" and followed by a parenthetical expression which may contain an argument to be passed to the function.

There are a number of functions which are included in FOCAL, such as:

FINT - returns the integer portion of a number
FABS - returns the absolute value of a number
FMEM - allows one to examine or deposit into a memory location.

FOCAL decides which function is being called by performing a "HASHING" of the function name and searching for that value in a function dipatch table. Using hash codes simplifies the lookup table design structure quite a bit. It may even speed things up a bit also.

If you wish to install your own functions, the hash code for the particular function name and the function address must be installed in the extra space provided in the lookup table.

Figuring out the hash code for your function is not so easy, however, unless you use FOCAL itself to do the computation.

In version 3D, place a BRK or JMP KIM at location $29EF. Then execute the following command:

```
SET X = F???(1)
```

where F??? is your new function name (FADC for example) and (1) is there because you need a parameter of some sort.

Program control will then be returned to KIM, or wherever your BRK vector pointed, and the hash code will be found in location $0065 as well as the Accumulator and the "X" register.

Several readers are preparing articles on FOCAL additions and modifications, so we have alot to look forward to in this section.

I just saw the latest Dr. Dobbs Journal at the newstand (computer store newstand, that is) and noticed that they published a rather large FOCAL program. (I don't recall the issue number).

Do YOU have any FOCAL articles or programs that you'd like to see in print? Then send 'em in.

I highly recommend the $12.00 FOCAL USER MANUAL from the 6502 Program Exchange to those who are learning to program in this language as well as those who are just curious and perhaps want to see how FOCAL compares to BASIC.

At the present time, FOCAL for the 6502 is available from two sources. Write to them for pricing and availablity.

ARESCO
PO BOX 43
Audobon, PA
19407

6502 Program Exchange
2920 Moana
Reno, Nevada 89509

basic

Microsoft a-little-over 8K Basic is available from two sources for about $100.

Johnson Computer P.O. Box 523 Medina, Oh 44256	Micro-Z Company Box 2426 Rolling Hills, Ca 90274

Both outfits are basically handling the same package except Micro-Z has added a facility to save data as well as programs through Basic and, has Hypertape built-in. I don't know if this increases the size of the Basic interpreter or not.

Neither of there two Basics is promable but Johnson Computer has indicated they have a promable version available for about $100. You have to give up SIN, COS, ARC and TAN though. This gets the size down to below 8K.

BASIC I/O MODS

Marvin L. DeJong
The School of the Ozarks
Point Lookout, Mo 65726

I had to agree with much of what Don J. Latham had to say about Microsoft Basic. The program modifies itself and that is a real pain as far as I am concerned, because if you want to do anything else, or if you blow something, you have to reload it. Once you get it running its nice, but I sure hate to sit around waiting and hoping for a tape to read.

Johnson Computer publishes some documentation. I wanted to convert Microsoft Basic to run on my KEM and MVM 1024 Video module, and without listings it can be difficult. For others who may want to sue Microsoft with a parallel ASCII keyboard and a CRT as opposed to a TTY system, you should be aware that changes must be made. I wrote a little routine following the suggestion of Gene Zumchak of Riverside Electronics, to find all the I/O locations in the program. For the 9 digit version these are:

```
INPUT   $2AE5 and $2456 (call KIM-1 input routine)
OUTPUT  $2A51           (call KIM-1 output routine)
```

which must be changed to call the users' own routines for his keyboard and CRT.

Also, there is a break routine at about $26DF. To be precise, address $26DF must be changed from 30 to 10. *(ED. NOTE:This mod did'nt work on my KIM)*

Now, if someone could tell me where to look to make the program list 16 lines of a program at once, rather than whizzing X number of lines past my CRT and showing me only the last 16, I would be grateful.

P.S. The Johnson Computer people have been very cooperative in working out some of my tape problems with the BASIC tapes.

A BASIC QUESTION

from the Editor

Does anyone know how to make Basic always come up in the SIN, COS, & TAN mode without having to answer the question with a "Y" everytime? As you may know, if you plan on saving programs, BASIC must be in the same mode when you load a program as when you saved it.

****

Got a note from Joe Donato, (193 Walford East, Sudbury, Ontario Canada P3E 2G8) who says he has subroutine for KIM 9 digit Basic (Microsoft) which permits the user to store programs on tape using I.D. numbers (Basic doesn't normally permit this). This subroutine contains HYPERTAPE and runs from 02EF to 03E3. It is available from Joe for $4.00.

BASIC TIMING & COMMENTS

F. E. Kempisty
1149 Garner Ave.
Schenectady, NY 12309

I have Johnson Computer's Microsoft 9-digit Basic. I disassembled it and found 2 extra commands GET and STEP which were not listed.

To speed up SAVEing programs. At location $275C change 4C 00 18 to 4C 00 02 and then locate Hypertape at $0200. Microsoft uses page one ($0100). My benchmark timing comparisons of Microsoft 9-digit Basic and Tom Pittman's Tiny Basic using the programs from Kilobaud #10.

	Microsoft 9-digit	Pittman's Tiny Basic	
			Without LET
Program 1 -	1.5 seconds	2 seconds	
Program 2 -	10.3 "	32 "	28 seconds
Program 3 -	18.5 "	51 "	46 "
Program 4 -	20.5 "	53 "	47 "
Program 5 -	22 "	62 "	
Program 6 -	31.7 "		
Program 7 -	49 "		

Speed is in the Top 5 - Good Huh!

KIM BASIC HINT

from the MICRO-Z CO.

The standard KIM BASIC will cause your BASIC program to stop running, and an "OK" to be typed, if you only hit "RETURN" in response to an INPUT query ("?"). Sometines, this can be annoying because the program must be re-run, or a "CONT" must be typed, if only the "RETURN" key is pressed inadvertently.

However, this can be changed by adding the following line early in your program:

```
XXXX POKE 10920, 169
```

With this change, when only the "RETURN" key is pressed in answer to a "?", a zero (0) is inserted in the variable and your BASIC program advances to the next line just as if you had entered a number. Of course, if you wish to leave your program in the middle, and go back to BASIC, you can always press the "ST" button on the KIM and re-enter through the "warm start" location (press space bar to get 0000 4C, then press "G").

Incidentally, don't forget to insert the following command near the end of the program, to put BASIC back the way it was:

```
YYYY POKE 10920, 165
```

The first command inserts and A9 (169 decimal) into location $2AA8 Hex (10920 decimal), and the second command replaces the A5 (165 decimal) that was originally there.

Since I last wrote you, we have modified our BASIC "DATA/SAVE" commands to record and playback both 'strings' and 'data' that are inserted while running a program. Previously, we only recorded the numerical data.

We are in the process of contacting all those who purchased our BASIC and are supplying them with the updated commands. However, I would appreciate a note in the USER NOTES: 6502 - asking those who purchased their BASIC from us to write Micro-Z, Box 2426, Rolling Hills, Ca 90274 - if they haven't heard from us as yet.

These commands will only work with the Micro-Z version of the Microsoft KIM BASIC so it would not be of any value to those who purchased their BASIC elsewhere. We are thinking of providing a package of the User Manual and a cassette of the added data for those who are not using our BASIC-but am not sure if anyone would be interested.

BASIC RENUMBER PROGRAM from

Harvey Herman
2512 Berkley Pl.
Greensboro, NC 27403

The following BASIC renumbering program may be of interest to your readers who use Microsoft 8K BASIC (9 digit version) on KIM. It is an adaption of a PET program which appeared in PET User Notes (Vol. 1 #5, July/Aug. 1978). RENUMBER, as it's name implies, renumbers the current program in memory. It converts both statement numbers and references in GOTO, GOSUB, and THEN statements.

The program can be utilized in several different ways. It can be loaded before beginning program development (an example follows). It can be loaded at any time using paper tape, if available, or even by hand (ugh!). It can be appended at any time if BASIC is modified (send SASE for details). I have written the program so that only the renumbered program remains after running. If this is not desired eliminate the POKES in lines 63950 and 63955.

The program has one restriction which I am aware of. When the new number in a reference has more digits than the old one, the first character or token preceding the old number will be replaced by the first digit of the new number. Line numbers present no problem as they always occupy two bytes. However, line number references have no specific number of characters (one thru five) and it is possible that tokens or commas, not spaces will be replaced. Where this occurs the lines will have to be manually re-entered. Remember to leave some space in front of each line number references and you won't have any problem. (see the example).

I have used another program which has no restrictions but requires a paper tape punch for intermediate storage of the renumbered program. Since many people are using KSR type terminals, without punches, I felt the following program would be more useful.

Program Notes (underlined letters were typed by the user):

1. DIM L% can be decreased if space is a problem. Decimal 120/121 (hex 78/79) is the pointer location for the start of BASIC programs in this version of BASIC.

2. L% (0) was used to store the new pointer to the end of BASIC programs as it was not "lost" after the first POKE in 63955.

3. The renumber program is deleted by POKE ing)'s at the end of the renumbered program.

4. The last two POKE's reset the pointer to the end of the renumbered BASIC program. CLEAR resets all other pointers.

5. In this example we see that the pointer at hex 7A/7B was set properly to hex $4078, two more than the 3rd zero at the end of the renumbered program.

[Editors Comment - Mr. Herman also mentioned that he has put together an enhancement package for 9 digit MicroSoft KIM BASIC which includes fast save & load, a real time clock, the GET command, paper tape control etc. He's asking $15.00 for the package or a SASE for more details. I'll review it for the next issue.

```
 ØK        RENUMBER
 LØAD

 KIM
 0000 4C G
 LØADED
 LIST

①  63900 CLEAR:DIML%(500):DEFFNR(X)=PEEK(X)+256*PEEK(X+1):L=FNR(120)
   63901 INPUT "STARTING LINE # AND INCREMENT";ST,IN
   63902 DEFFNM(X)=INT((ST+IN*X)/256)
   63905 N=FNR(L):X=FNR(L+2):IFX<63900THENA=A+1:L%(A)=X:L=N:GØTØ63905
   63907 EN%=L:EH%=INT((L+3)/256):EL%=L+3-256*EH%
②  63908 L%(0)=EH%
   63910 L=FNR(120):FØRB=0TØA-1:N=FNR(L):PØKE(L+3),FNM(B)
   63912 PØKE(L+2),ST+IN*B-256*FNM(B)
   63915 F=0:FØRC=L+4TØN-1:P=PEEK(C):IFP=136ØRP=140ØRP=161THENF=1:GØTØ6395
 0
   63920 IFF<>0THENIFP>47ANDP<58THEND=10*D+P-48:G=G+1:GØTØ63950
   63925 IFF=0ØRD=0GØTØ63950
   63930 FØRE=1TØA:IFD=L%(E)GØTØ63940
   63935 NEXTE:D=0:G=0:GØTØ63950
   63940 D=0:E$=STR$((E-1)*IN+ST)+"    ":H=LEN(E$)-4:C=C-G:IFH>GTHENC=C-1
   63942 IFH>GTHENG=H
   63945 FØRI=1TØG:PØKEC,ASC(MID$(E$,I+1,1)):C=C+1:NEXTI:G=0
③  63950 NEXTC:L=N:PRINTB;:NEXTB:PØKEEN%,0:PØKE(EN%+1),0
④  63955 PØKE122,EL%:PØKE123,L%(0):CLEAR:END
 ØK
 10 REM
 20 GØTØ 10
 30 GØSUB 20
 40 ØN A THEN 10, 20, 30
 50 END
 RUN 63900
 STARTING LINE # AND INCREMENT? 100,10
  0  1  2  3  4
 ØK
 LIST

  100 REM
  110 GØTØ100
  120 GØSUB110
  130 ØN A THEN100,110,120
  140 END
 ØK
```

```
    KIM
    0000 4C 7A
⑤  007A 78
    007B 40 4073
    4073 80
    4074 00
    4075 00
    4076 00
```

tiny basic

TWO TINY BASIC MODS

Michael E. Day
2590 DeBok Rd.
West Linn, Or 97068

Tom Pittman's TINY BASIC TB651K V.1K may have a bug!!!

The following program has the ability to lock you out of your computer:

```
1 RUN
```

What happens, is that when you type RUN, TINY begins execution, and the first statement it sees is RUN; which causes TINY to begin execution again. During all of this there is no test for a BREAK, which leaves the computer running away happily ignoring you.

This is no big deal, unless your computer happens to be located in a remote location (Like across town!), then it becomes a pain.

I found this bug late one night when nothing else was going right, (MY keyboard has not been the same since) and I typed it in by mistake.

Normally, I wouldn't care about it, but due to the circumstances it 'bugged' me, so I decided to do something about it. The following is the cure, and is located in the execute routine (XQ).

```
053F  A5 2A     LDA  2A     Get IL pointer (ADL)
0541  85 C4     STA  C4     Save it
0543  A5 2B     LDA  2B     Get IL pointer (ADH)
0545  85 C5     STA  C5     Save it
0547  4C 0F 05  JMP  050F   GOTO NX routine
054A  EA        NOP         Not used
054B  EA        NOP         Not used
```

This replaces the previous data, and allows a break test on execution.

The multiple statements per line modifications consists of changing the address of the Branch End routine to the new address, changing the name of the old NX IL code to NS (address remains the same), and the addition of the new NX IL code and address. NX retains the old meaning and description of Next Line. The new NS code searches for the Next Statement by looking for a colon (:) or carriage return, and passing control depending on what it has found.

The ML routine for the NS code is a modification of the old NX routine with a subroutine located at $0AE8. This routine causes execution of the next statement if a colon is found, it goes to the next line if a carriage return is found and in the run mode, otherwise it returns to the command mode.

The new ML routine for the BE code tests for a carriage return or colon to indicate statement end.

A modification to the IL is needed at $09B4 in order to use the colon (:) as a terminator, as this character is used to produce an X-OFF (DC3) after a print statement. This is modified to produce the X-OFF on an exclamation point (!) instead.

Another modification to the IL must be made at $09F5. This is required to make TINY begin execution on the next line rather than next statement following GOSUB RETURN. This is required due to the fact that TINY only remembers the line number for the return link, so if the GOSUB was not the first statement in the line, a hard loop would be set up. With this modification however, execution will begin on the next line, and not the next statement after a GOSUB has been executed.

A modification is made to the IL at $0A26 which causes execution to begin on the next line after a REM statement instead of beginning with the next statement. This allows colons to be in REM statements. It allows for more powerfull IF THEN statements. I.E.: IF A=0 THEN REM: LET A=1: PRINT A,: GOTO 20. In the above example if A is equal to 0, then execution begins on the next line, otherwise the rest of the present line is executed.

The colon may not be used in a print statement that is the second part of an IF THEN statement, since if the test is not true, then a search for the next statement is begun, and termination of the search will be prematurely done upon deteciton of the colon in the print statement. The colon may be in any other print statement however, even on the same line as the IF THEN statement. It just can not be used as the second part of an IF THEN statement.

The GOSUB will always be the last statement executed in a line. I.E.:
IF A=0 THEN GOSUB 20: LET A=1: PRINT A: GOTO 10
In the above example if A is equal to 0, then the GOSUB 20 is executed, and execution continues with the next line following the example upon RETURN from the GOSUB. If A is not equal to 0, then the GOSUB is skipped, and the rest of the line is executed.

IL ADDRESS CHANGES

```
CHANGE TO      WAS

022C  F2       FD
022D  0A       03  Branch End (BE)
025A  E0       9F
025B  0A       05  Next Line (NX)
```

Old IL code NX now becomes NX (Next Statement) there is no address change however.

IL ROUTINE CHANGES

```
09B4  83  A1   !          X-OFF On (!) exclamation
          (3)  BC 09B8    point instead of (:) colon
09F5  1E       NX         NX on Return instead of NS
```

ML ROUTINE ADDITIONS

NEW NX ROUTINE

```
0AE0  20 14 04  JSR 0414  Search for "CR"
0AE3  D0 FB     BNE 0AE0  Con't until found
0AE5  4C 0B 05  JMP 050B  Get new line
```

NEW NS ROUTINE

```
0AE8  20 14 04  JSR 0414  Search for terminator
0AEB  F0 04     BEQ 0AF1  Return if "CR"
0AED  C9 3A     CMP #3A   Return if ":"
0AEF  D0 F7     BNE 0AE8  Otherwise try again
0AF1  60        RTS
```

NEW BE ROUTINE

```
0AF2  20 25 04  JSR 0425  Read BASIC character
0AF5  C9 0D     CMP #0D   If it is a "CR"
0AF7  F0 F8     BEQ 0AF1  Return
0AF9  C9 3A     CMP #3A   or a ":"
0AFB  F0 F4     BEQ 0AF1  Return
0AFD  4C 64 03  JMP 0364  Otherwise go branch
```

ML ROUTINE CHANGES

NS ROUTINE

```
0506  20 E8 0A  JSR 0AE8  Find terminator
0509  B0 0C     BCS 0517  End line?
050B  A5 BE     LDA BE
050D  F0 23     BEQ 0532  Run mode?
```

RAMBLINGS ABOUT PITTMAN TINY BASIC by

Lew Edwards

Bought Tom Pittman's TINY BASIC, also his "Experimenter's Kit". Perhaps you might be interested in the following comments.

Things "not in the book" or at least not too clear.

Saving and loading basic programs using KIM cassette routines---Use the values in $0020 & $0021 for SAL & SAH and use the values in $0024 & $0025 for EAL & EAH when dumping to cassette. When loading the saved programs, transfer the values in $17ED & $17EE to $0024 & $0025 and enter TINY via the "warm start". Of course before loading the tape, you should have previously done a "cold start" to initialize the basic pointers, etc. Expect your whole system to crash if you try to make program changes without setting 24 & 25 to the correct values. You can append a second program to the one in memory if the second program has line numbers higher than the first. I have written a line renumbering program if anyone is interested. The second program is loaded in starting at the address in $0024 & $0025 minus four. Again, transfer values from $17ED & EE to $0024 & 25. I am using a tape loading subroutine callable as a USER function, which directly uses 24 & 25 as a pointer for storing recovered data so that it is automatically set up as end pointer for user programs.

HOW TINY STORES PROGRAMS:

User programs start at the address stored in $0020 & $0021 and lines are stored exactly as entered from the keyboard. The line number is stored as two hex bytes, all the rest as ASCII, ending with the carriage return, 0D(hex). All lines are stored in sequence as numbered, with TINY doing the editing as each line is entered (or deleted, or replaced). TINY stores a ZERO line number in the two bytes follwoing the CR in the last line of the program. When TINY responds to a CLEAR command, it puts the zero line number in the first two bytes of the user program space and initializes the pointers. If you should accidently clear, say be using the "cold" start to re-enter basic, after having entered a program; you can salvage the program by loading a value in the first byte of user memory equal (in hex) to the original line number of the first line. Of course, if the number is over 255, you'll have to put the high order value into the second byte. This will let you list and run the program, but if you want to make any changes, you'd better restore the pointer at 24 & 25. You can search through memory to find the right address using the following rules. First, line numbers are contained in the two bytes immediately following a carriage return (0Dhex). The last CR is followed by two zero value bytes. Add 5 to the address of the last CR and load the result into 24 & 25.

MACHINE LANGUAGE SUBROUTINES:

These can be used by calling a USER functions. If you want an ML subroutine to be included with your TB program, it can be "contained" within REM statements placed after the last line of your program. Make one or more REM statements using enough characters between the first REM and the last CR to accomodate your subroutine. The result will be garbage on a LIST, but that's immaterial. The ML subroutine can then be called by: X=USR(USR(S+20, 36)+USR(S+20,37)*256-n) where X is the result returned from the subroutine in the A & Y registers, S is the starting address of TINY BASIC, and n is the number of bytes reserved for the machine language code +6. If the ML subroutine is to be called more than once, a variable may be set to the value within the opening and closing parentheses. Second and third arguments may be included to pass parameters. The line renumber program I wrote in TB uses this technique to locate the line numbers. I had at first written it using only the TB built in USER routines for "peek" and "poke", but it ran too slowly to suit me. No, the renumber program does not renumber the goto's and the gosub's.

****

forth

We're going to be hearing alot more about 6502 FORTH. Looks like an ideal "hackers" language.

That rumor I mentioned in #12 about there being a FORTH User Group newsletter was true. I've received two issues of FORTH DIMENSIONS and they are quite informative. You won't believe how simple it is to make FORTH understand German.

FORTH DIMENSIONS is available for $5.00/6 issues from: Forth Interest Group, 787 Old County Rd., San Carlos, Ca 94070.

I've been informed that the Decus Forth manual now costs $12.00. This manual provides the most implementation information I've found yet. Order manual 11-232 from DECUS, 126 Parker St., Maynard, Ma 01754.

There seems to be a bit of controversy growing over some of the versions of FORTH which are beginning to crop up. This controversy purportedly stems from the contention of the FORTH Interest Group that some of the languages which use the name "FORTH" don't implement all FORTH features.

According to FIG, complete versions of FORTH should contain:

1. indirect threaded code
2. an inner and outer interpreter
3. standard names for 40 major primitives
4. words such as; CODE, BLOCK, DOES>, (or ;:), which allow increased performance.

FORTH is especially useful for real-time control-type applications. Some of the programming examples I have seen indicate that programs tend to get very modular and structured because of the way FORTH operates.

At the present time, there are at least 6 implementations of FORTH, or "forms" of FORTH on the 6502. As for as I know, most of the 6502 FORTH implementations were done independently which indicates tremendous interest in this language from the 6502 fraternity.

We'll be trying to keep up with this "FORTH" explosion and will report on these different implementations when/if they become available.

FORTH COMMENTS & EXAMPLE

John P. Oliver
P.O. Box 12248
University Station
Gainesville, Fl 32604

I am currently running FORTH from Programma Consultants, 3400 Wilshire Blvd., LA, in my PET 2001. I am convinced that FORTH is the ideal language for the hardware havker who needs to be able to program drivers for interfacing. FORTH is an interpreter, compiler, and assembler all at the same time. You can use a higher level language to do loops, blocks, arithmetic, and imbed assembler code or machine code at any point. Normal FORTH code runs about 50% of the speed of optimum machine code...much faster than BASIC. If FORTH is too slow in some time critical routine, simply imbed machine code. The following is an example of a FORTH program to point a telescope, select a blue filter, make a 20 second integration of the star brightness, store the result in the array DATA, and print the result:

```
12 30 15 RA +72 36 12 DEC POINT BLUE FILTER
20 SECS  INTEGRATE DUP DATA ()= PRINT
```

I have routines in FORTH which access the PET internal clock so that I can store the time and/or print it out etc. I expect to implement FORTH in my AIM-65 as soon as it arrives...I probably will not put FORTH in any of my KIM systems since it is not well adapted to the keypad and hex display of the KIM.

KIM users who read a little german should be aware of the 65xx Micro Mag published by Roland Lohr, Hansdorfer Strasse 4, 2070 Ahrensburg, West Germany. The first two issues have had a lot of very good systems software for KIM systems. Relocatable loaders, fancy tape operating systems etc. The price appears to be DM 46 by surface mail to the US but Herr Lohr may also have an airmail rate for US. Amazingly, both issues have had discussions of the AIM-65 although I have not yet seen a discussion in a US hobby magazine.

assemblers

TWO PASS PATCH TO ARESCO ASSEMBLER

John Eaton
1126 N 2nd St.
Vincennes, In 47591

Here is a patch to the ARESCO resident assembler that will convert it into a two pass assembler. This change will give you source listings that contain all the program addresses (no**) and will even make the object code more efficient. The patch consists of the following code:

```
257A  4C F0 30
30F0  B1 52 A0 03 29 1F C9 10
      D0 01 88 A9 01 4C 7D 25
```

This will work in the version assembled at address $2000. You will have to change the absolute addresses if you have the version at $E000.

To use this patch you assemble a source program using the "A" command as normal. The printed listing you receive will probably contain several errors. That's all right since all the run did was to set up the symbol table. Now rerun the same program again starting the assembler at the warm start address of $2011. This time you will get a correct listing with no **'s. You may also notice that you can also use forward referencing with arithmetic operations which you can't do with the original assembler.

The second pass through the assembler does not reset the error count or reenter any of the OPT's so if you want to disable the first printing and enable the second one then you have to manually reset the flags that were set in page zero.

Also, if you have the $2000 version of the assembler then change address $321A to 50 and $258D to 2D. That prevents the assembler from printing the line numbers as one plus their actual value.

MODS TO THE MICRO-SOFTWARE SPECIALISTS ASSEMBLER

Richard M. Bender
Box 276 RD 1
Ebensburg, Pa 15931

I am successfully using MSS's resident Assembler/Text Editor (ASM/TED). There is no comparison between this program and their previous release as many of you may have experienced. Since I only have a TVT and no hard copy output I did have some difficulty in correcting errors in my source programs because of the lack of the ability to list single lines only. However with addition of the routines listed on the next page the ASM/TED will now have this capability. One precaution--be sure to define the symbol table and text file sufficiently above the end of the ASM/TED program to allow room for the added routines.

NOTES: This addition for the TED/ASM that is loaded from $2000 up. For other locations you will have to change line numbers 0100, 0120, 0250, 0270, 0300, 0320, 0460, 0470 and 0570 for proper relocation.

I would recommend that lines 0100 thru 1030 be omitted until the assembly completes successfully, with no errors, then enter these four lines and reassemble. During assembly the object code is generated and LOADED into memory even though errors may be detected. Obviously an "R" command is not needed. Happy assemblying.

(listed in TED/ASM format)

```
0100      .OR 26F7       area in TED/ASM to be changed
0110       JSR NUMB       extract desired line number
0120      .OR 26AE       another patch
0130       JMP REQT       is current line the desired one?
0200 :
0210 :  ROUTINE EXTRACTS NUMBER FROM LINE REQUEST
0220 :  IN  LIST COMMAND
0230 :  (e.g. to list line 142, type  L0142  and carriage rtn)
0240 :
0250      .OR 282D       end of TED/ASM program
0260 NUMB LDY 01         get first two digits of line number
0270       JSR $245B      convert to packed decimal form
0280       STA $0109      save in TED/ASM line buffer
0290       INY            get pair of digits
0300       JSR $245B      and do the same conversion
0310       STA $0108      -into line buffer also
0320       JMP $269F      back to normal LIST command processing
0400 :
0410 :  ROUTINE COMPARES CURRENT LINE WITH DESIRED LINE
0420 :  NUMBER AND IF THEY MATCH, PRINT THIS LINE
0430 :
0440 REQT LDA $0104      does a requested line number exist
0450       BNE HERE       branch if yes
0460 PRNT JSR $27FA      no, print this line after a CR/LF
0470       JMP $26B1      to normal list operations
0480 HERE LDY 00         lets match line numbers
0490       LDA (LOCA),Y   get current number from TED file
0500 LOCA .DL 00D5       TED file base address stored here
0510       CMP $0108      against desired line number (2-digits)
0520       BNE NOPE       no match
0530       INY            yes, check next 2-digits
0540       LDA (LOCA),Y
0550       CMP $0109      do they match too?
0560       BEQ PRNT       yes, go CR/LF and print line
0570 NOPE JMP $26CF      no, go get another line in TED/ASM
0589 PGEN .EN            that's it!
```

design corner:

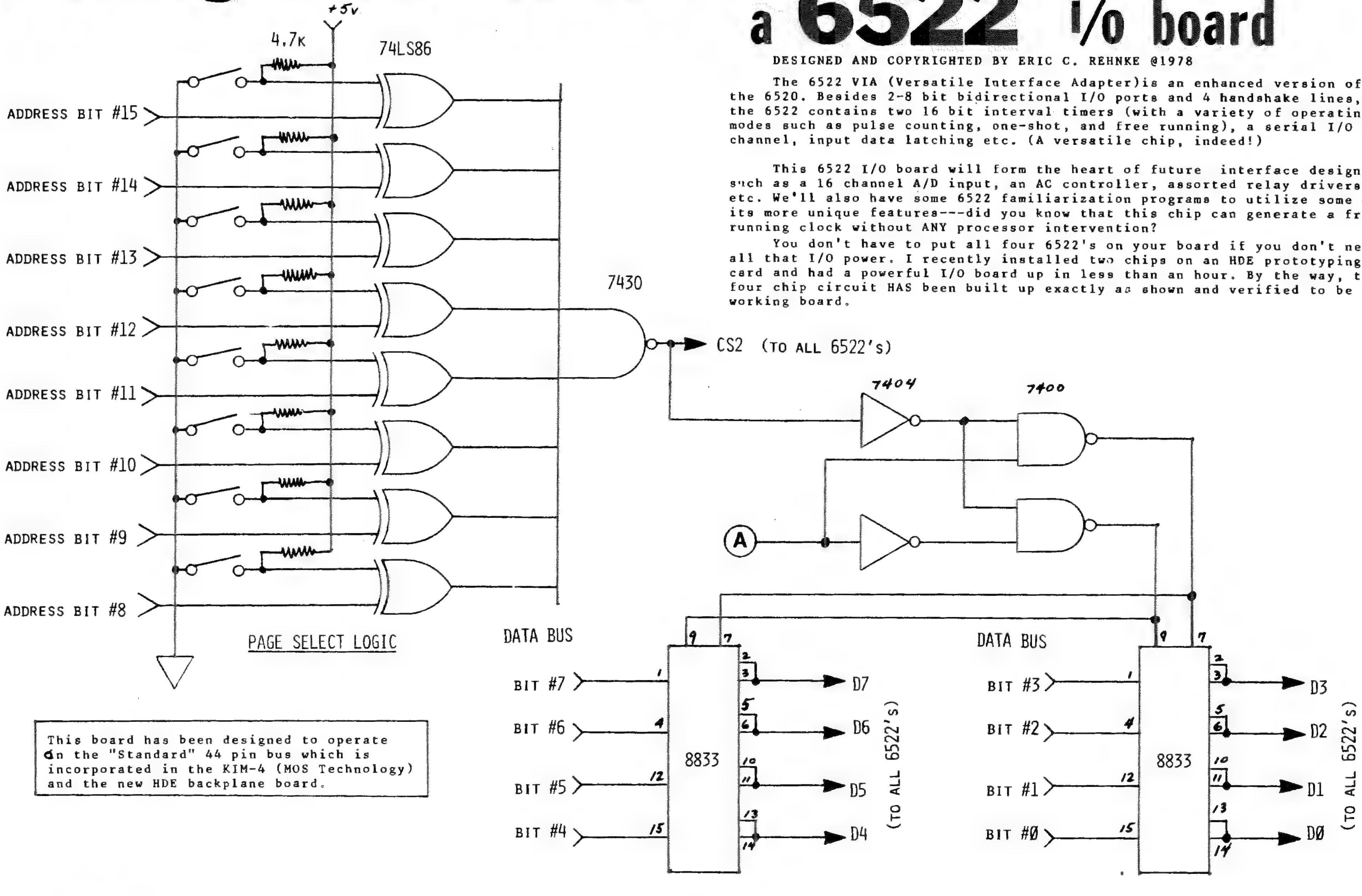
a 6522 i/o board

DESIGNED AND COPYRIGHTED BY ERIC C. REHNKE @1978

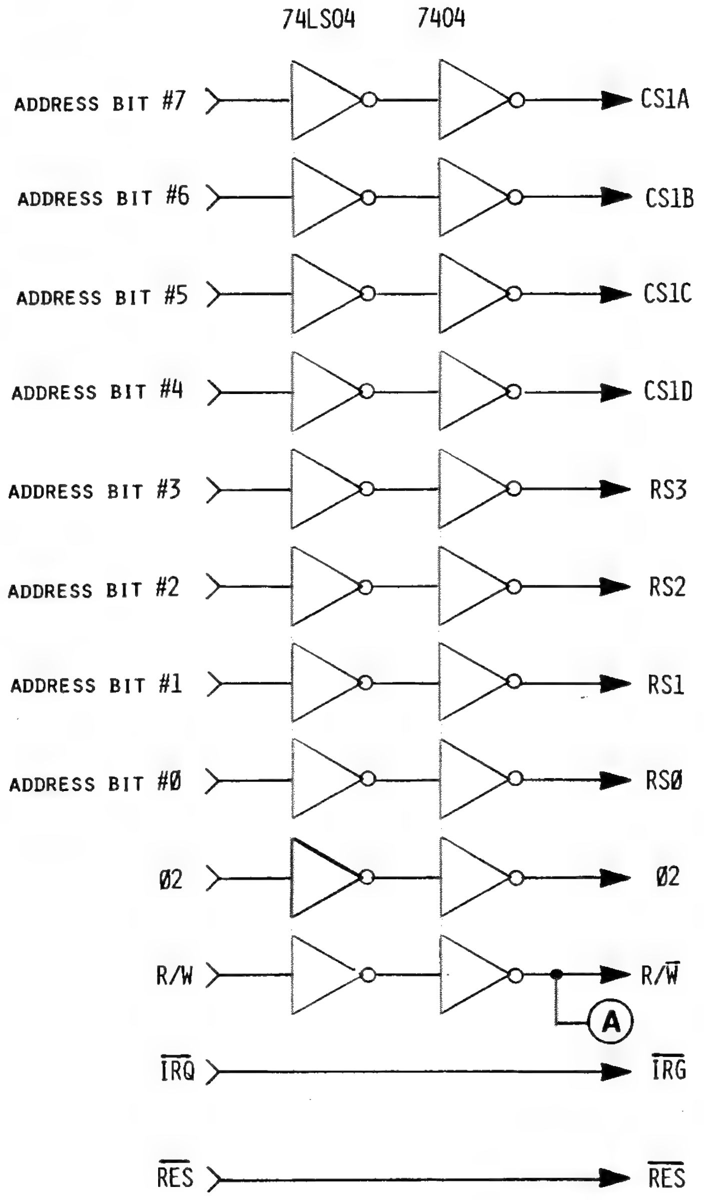
The 6522 VIA (Versatile Interface Adapter)is an enhanced version of the 6520. Besides 2-8 bit bidirectional I/O ports and 4 handshake lines, the 6522 contains two 16 bit interval timers (with a variety of operating modes such as pulse counting, one-shot, and free running), a serial I/O channel, input data latching etc. (A versatile chip, indeed!)

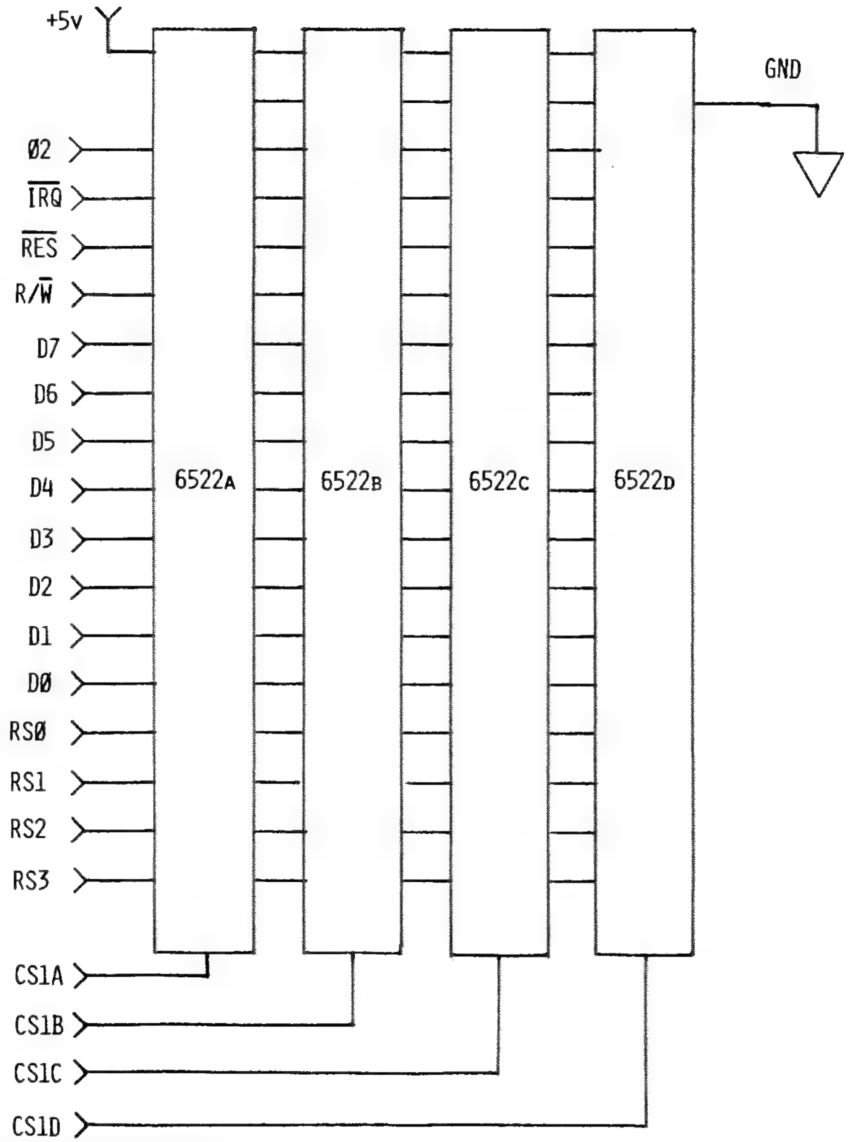
This 6522 I/O board will form the heart of future interface designs such as a 16 channel A/D input, an AC controller, assorted relay drivers etc. We'll also have some 6522 familiarization programs to utilize some of its more unique features---did you know that this chip can generate a free-running clock without ANY processor intervention?

You don't have to put all four 6522's on your board if you don't need all that I/O power. I recently installed two chips on an HDE prototyping card and had a powerful I/O board up in less than an hour. By the way, the four chip circuit HAS been built up exactly as shown and verified to be a working board.



This board has been designed to operate on the "Standard" 44 pin bus which is incorporated in the KIM-4 (MOS Technology) and the new HDE backplane board.





ADDRESSING THE BOARD:

As configured, each 6522 will be accessed as follows:

6522A=$XX80-$XX8F   6522B=$XX40-$XX4F   6522C=$XX20-$XX2F   6522D=$XX10-$XX1F

(XX is determined by the page select logic) Care must be taken that only one address line in the bit4-bit7 group be high at any one time or more than one chip will be accessed. Of course, this caution need be taken only when the board is selected.

KIM-4 BUS PINOUT

COMPONENT SIDE			WIRING SIDE
GROUND	1	A	GROUND
SYNCH	2	B	ADDRESS BIT Ø
$\overline{\text{RDY}}$	3	C	ADDRESS BIT 1
$\overline{\text{IRQ}}$	4	D	ADDRESS BIT 2
-16 v. unregulated	5	E	ADDRESS BIT 3
$\overline{\text{NMI}}$	6	F	ADDRESS BIT 4
$\overline{\text{RST}}$	7	H	ADDRESS BIT 5
DATA BIT 7	8	J	ADDRESS BIT 6
DATA BIT 6	9	K	ADDRESS BIT 7
DATA BIT 5	10	L	ADDRESS BIT 8
DATA BIT 4	11	M	ADDRESS BIT 9
DATA BIT 3	12	N	ADDRESS BIT 1Ø
DATA BIT 2	13	P	ADDRESS BIT 11
DATA BIT 1	14	R	ADDRESS BIT 12
DATA BIT Ø	15	S	ADDRESS BIT 13
BDSEL *** (N/C)	16	T	ADDRESS BIT 14
+16 v. unregulated	17	U	ADDRESS BIT 15
$\overline{\text{DMA}}$	18	V	Ø2 clock
+8 v. unregulated	19	W	R/$\overline{\text{W}}$
+8 v. unregulated	20	X	$\overline{\text{Ø2}}$ clock
+5 v. *** (N/C)	21	Y	+5 v. *** (N/C)
GROUND	22	Z	GROUND

Here it is! The 44 pin standard "KIM-BUS". This bus definition is applicable to the KIM-4 from MOS Technology as well as the new backplane board from HDE Inc.

Pins 16, 21, and Y have been left uncommitted on all present KIM-4 boards as these signals (+5 and BD select) were used ONLY when a single board was added to the KIM-1 without the motherboard. These pins DO need to be defined as busable signals before signal incompatiblity problems arise as did with the S-100 bus. I feel that clock Ø1 should be added to ease the problem of adding dynamic RAM to the system and perhaps the remaining two signals could be used for some sort of interrupt daisy-chain (like on the PDP-8 or 11 bus. Thats my idea.

WHATS YOURS???????

According to sources at MOS Technology, a keyway will be installed between pins 18 and 19 to eliminate the possibility of plugging boards in backwards. (great idea!)

VIDEO & GRAPHICS

VIDEO DISPLAYS: ERIC

STANDALONE vs MEMORY MAPPED

It seems that there are a number of us who have purchased memory mapped video displays such as Polymorphics VTI-64 or Processor Tech's VDM boards for one reason or another and are quite shocked to find they need to write some software to get the thing to talk to KIM. This is unfortunate as it makes for a very frustrating time. Perhaps we should talk about what the KIM can & cannot do in the way of peripherals. And what is needed to hook up to a video display device.

First of all, KIM is configured to communicate with a ASR-33 Teletype tm which has a 20 ma loop and talks serially (which means that the data bits march down the wire one after another). Of course anything else that can fake KIM into thinking it's a teletype will also work. This includes most standalone video terminals (such as made by Hazeltine and Lear-Siegler) and some other hardcopy terminals such as DECwriter etc. If it isn't serial and doesn't use a 20 ma loop - Forget it! You'll have to do sone converting to get your whatzit to talk to KIM. Oh yeah, your whatzit terminal HAS to speak ASCII.

A memory mapped video display, on the other hand, is a totally different animal! There's nothing parallel or serial about it. Except perhaps the fact that its got a parallel address & data bus. To the computer, the video board looks like a block of memory - NOTHING MORE!!

Some computers, such as PET, APPLE, and SOL, have programs built in to make these memory-mapped displays look like output devices - but the KIM DOES NOT! You would have to write programs to: clear the screen by initializing every screen location to an ASCII "space" character; form a cursor (usually a white square); and prudently place ASCII characters on the screen to make some sense; make sure the display does the proper thing in response to a carriage return; etc.

Not an easy task for most beginners!

And if you expect to be able to operate the KIM ROM monitor program from your memory-mapped display, FORGET IT!! There's no straightforward way to do it. You'd have to rewrite a completely new monitor program around your new display device.

Sound like alot of trouble? Youre right! If you arent prepared (or able) to write a complete new monitor from scratch, or perhaps modify an existing monitor, such as XIM (Pyramid Data System, 6 Terrace Ave, New Egypt, NJ 08533) to work with your display then I'd suggest you hold off this ambitious project at least until you can get some help.

On the other hand, memory mapped video displays are so much more versatile than serial displays that the extra trouble to bring up this type of peripheral may be worth the extra trouble to you if you are at all talented in the software dept. Immediate access to any position in the display area makes it possible to run real-time games such as Chase, animated LIFE generations, Breakout, Pong etc etc.- as well as performing double-duty as the more mundane video terminal style output device.

Other possibilities for memory-mapped video include split screen displays where you effectively have two output devices. In a 64x16 video board, you could have two 64x8 scrolling displays each fully independent of the other. This could, of course, be extended to provide a number of such "windows" on your screen. Remember the cliche "you're only limited by your imagination"? Well, it holds true here.

If you're not into fun and games, then how about a fancy string edition (word processor) which could immediately display updated text as fast as you can modify it?

I recently had an opportunity to play around with the "Electric Pencil" string editor (on an 8080 system) and really enjoyed watching the text file open up right on the screen when new characters were inserted and close up when characters were deleted. It must be seen to be appreciated.

By now, it should be apparent that there are tradeoffs involved in any decision and this one's no different. It is hoped that by now, you'll have some idea of the pros and cons of each approach to video displays, and will be able to make an intelligent decision.

Here are some companies that make/sell serial video displays: RAMSEY ELECTRONICS Box 4072P, Rochester, NY 14610 716-271-6487 (I've seen this 64x16 display running and can recommend it); OTTO ELECTRONICS P.O. Box 3066, Princeton, NJ 08540; MICRO-TERM INC P.O. Box 9387, St. Louis, Mo 63117 314-645-3656

COMMENTS ON THE "VISIBLE MEMORY" VIDEO BOARD

from Lew Edwards

I promised some comments on Hal Chamberlains "VISIBLE MEMORY" which I have running as a video display along with a parallel keyboard. The board (Micro Technology Unlimited K-1008) has worked perfectly since I first hooked it up back in April. The documentation is excellent as is the software for it from MTU. The VM board includes decoding to enable the KIM interrupt and reset vectors, so you dont have to worry about that incase you want to add more memory.

Had to wait about 5 weeks for the software, but that's no problem now. The software package is a beautiful job also. The only drawback was that I had to program around the ROR instruction.

Wrote a HEX dump routine that prints a page at a time on the screen and am using Steve Wozniak & Allen Baum's disassembler as published in Dr. Dobb's in Sept. '76.

TVT-6 ADVENTURE

Dennis Chaput
Rockwell Hobby Computer Club
12851 Olive St.
Garden Grove, Ca
714-892-2703

I ordered my TVT-6 last October and received it about ten days later. The assembly instructions were a re-print of the article in "POPULAR ELECTRONICS", where I first read about the TVT-6. The unit went together OK. I mounted a 16 pin IC socket near C15 on the KIM-1 for the RAM connections so that I could completely disconnect the KIM module.

I loaded the 16 line by 32 character per line program and started the program. My old PENNY-CREST tube type TV came alive and locked-in without any adjustments. I have an RCA phono jack mounted on the rear cover with a slide switch to select TV or external input.

Problem number 1. The right hand column of dots were missing on every character. I had to change C5 to a smaller value. It took about three tries before it worked right. Using an ASCII to HEX chart and the KIM keyboard, I could load any character I selected. Now I was ready for the cursor program.

Problem number 2. I loaded the 16 by 32 Full Performance Cursor Program and the IRQ interrupt vector address. With my ASCII keyboard connected, I was ready, but my system wasn't. No matter what I did, I could not interrupt KIM to get into the cursor program. After much time and in desperation I wrote a long letter to PAIA Electronics explaining everything. I received a two line reply to enter the interrupt vector at 17FE-00 and 17FF-01. I had entered it at least 100 times before.

I know very little about programming but can usually follow through one copied from "KUN" or "FIRST BOOK OF KIM", so back to the books. I ran across a code "CLI 58" Clear Interrupt Disable Bit. I hunted all through the program but could not find one. I changed address 1780 NOP EA (two cycles) to 1780 CLI 58 (two cycles) and that fixed it. I've had no problems with IRQ interrupt since.

Problem number 3. This one was much easier. The Erase to End of Screen (EOS) was dead. I changed the following address: 013E CMP C9 13 to 013E CMP C9 12 and now it worked. This also made it agree with the chart at the top of the page. Also note that "SPARE HOOK" is also wrong. It should be: 013A CMP C9 13.

Problem number 4. SCROLL UP would not work properly. It worked just enough to show that it was close. It would give multiple cursors, repeat lines, modify characters and worst of all would "blow the program" very often. Change the following: 0184 LDA A9 01 to 0184 LDA A9 03, now it works OK.

I imagine these problems would be obvious to most people but were very difficult for me. Maybe they will help another beginner. KEEP UP THE GOOD WORK.

TVT-6 RAM EXPANSION

Michael Allen
6025 Kimbark
Chicago, Il 60637

My article in User Notes #12 has prompted a number of inquiries regarding how I added S.D. Sales 4K memory board to KIM/TVT-6. The following description is how I did it. I stress the I because I am not a hardware expert. There may be a better way and I am unsure how to add more memory for addresses above 2000H.

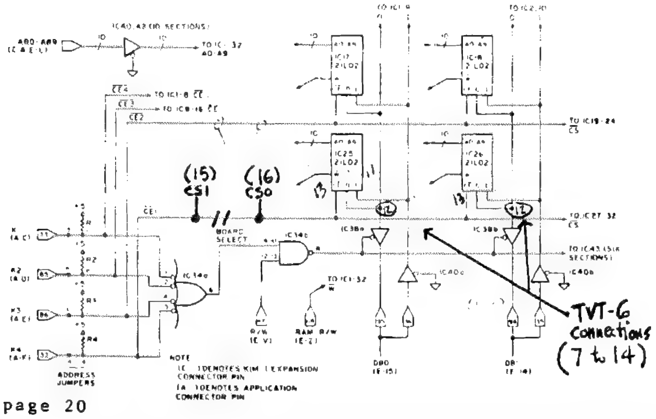
The S.D. Sales 4K board should be modified exactly as per the article in Kilobaud #4, KIM-1 Memory Expansion, by: Bob Haas. - Except the jumper between IC37 (pin 5) and IC33 (pin 6) should be omitted. Instead connect a jumper from TVT-6 (pin 15) to IC37 (pin 5) and a jumper from TVT-6 (pin 16) to IC33 (pin 6). (Of course it is best to bring these connections to an unused pin on the RAM board edge connector to avoid direct jumpers.)

Then TVT-6 pins 7 through 14 (VD7-VD0) should be connected to each pin 12 of IC's 25 through 32 on the RAM board. This is most easily accomplished as tabulated below:

TVT-6 Contact:	4K RAM PIN:
7 (VD7)	PIN 14 IC43
8 (VD6)	PIN 12 IC43
9 (VD5)	PIN 4 IC43
10 (VD4)	PIN 6 IC43
11 (VD3)	PIN 2 IC43
12 (VD2)	PIN 10 IC43
13 (VD1)	PIN 14 IC38
14 (VD0)	PIN 12 IC38

All other connections between TVT-6 and KIM should be as per TVT-6 instructions, including the removal of the ground connection to Application connector contact K (decode - enable).

Fig. 1 is from Bob Haas' article:



POLYMORPHICS VIDEO BOARD MODS

ERIC

VTI-64 owners! Now you can have your cake and eat it too!

After using the Poly video board for a time it became obvious that for some applications it would be nice to have the facility for reverse video in lieu of the graphics capability. As you know, the normal Poly board uses bit 7 to choose between alphanumerics and graphics. (The polarity of bit 7 is reversed from normal also, but we'll discuss that later).

Fortunately, the solution to the problem was already at hand. It was in the form of a piece of documentation from the Solid State Music video board. This board uses a similar method of generating graphics as the Poly card but also included the option to change from graphics mode to reverse video mode by the flick of an on board switch.

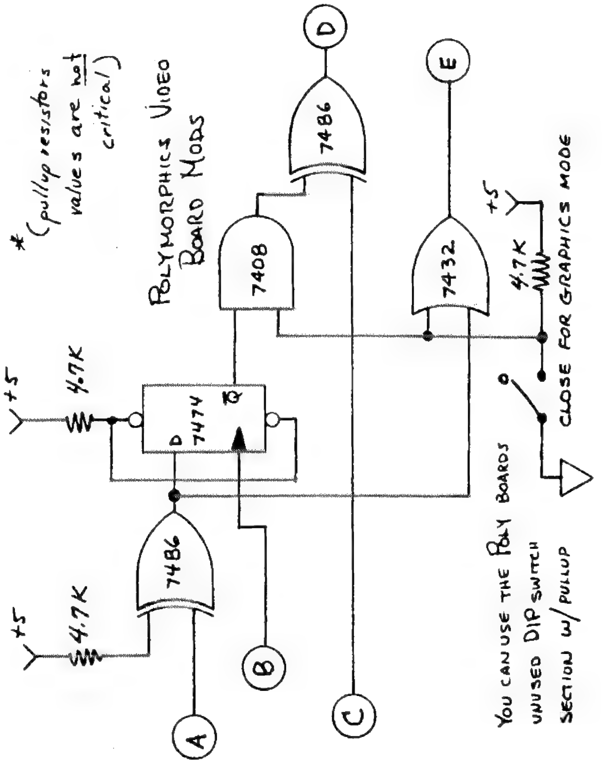
It then became a simple matter to transfer the mode switching logic over to the Poly board, and that's just what I did.

Two traces on the board must be cut. One that goes from the data latch IC40 (74273) pin 19 to the data multiplexers IC 33 & 36 pin 1 and the other one which runs from the shift register IC35 (8274) pin 6 to the video output buffer IC 31 (7407) pin 9.

The circuit below is then connected to the Poly board (Rev F)
A connects to IC40 (74273) pin #19
B connects to IC35 (8274) pin #7
C connects to IC35 (8274) pin #6
D connects to IC33 (74LS157) pin #1
E connects to IC31 (7407) pin #9

This mode also corrects Poly's design "accident" of needing bit 7 set to "1" for normal ASCII and set to "0" for graphics. If you don't want to modify your board for the reverse video option but still want to have bit 7 act normally - no sweat. On the Rev F board there is a spare gate in U5 (74LS132) that can be used to invert the signal coming from IC19 (74273) to IC33 and 36 pin 1. (Don't forget to cut the trace).

Next issue will have a mod to adapt the Poly Video board to the KIM bus. It's not as easy as you might think, but, thanks to one of our readers, I now have the board running in my KIM-4 system.


*(pullup resistors values are not critical)
POLYMORPHICS VIDEO BOARD MODS
D
E
7486
7408
7432
7474
7486
+5
4.7K
+5
4.7K
+5
4.7K
Q̄
D
CLOSE FOR GRAPHICS MODE
YOU CAN USE THE POLY BOARDS UNUSED DIP SWITCH SECTION W/ PULLUP
A
B
C

debug

SLOW STEPPER IV

by Lew Edwards

Slow stepper will automatically step through a program in the same manner as KIM's SST routine. The stepping rate is under keyboard control, and the program may be stopped at any point for examination and modification of registers, flags and memory data. Slow stepper is relocatable if the NMI vector is re-directed to the new NEXT location by changing locations 0100 & 0105 to the proper values. Slow stepper will fit into the RAM area starting at 1780 if the initialization of 0100 to 0113 is performed manually or separately, and the program is loaded starting at HALT.

To use SLOW STEPPER, PB7 (A-15 terminal) must be jumpered to NMI (E-6 terminal). To run SLOW STEPPER, enter the starting address 0100, press "GO", display remains unchanged. Enter the starting address of the program to be executed, press "ST" key and after a short delay the program will begin to run at the slowest speed. Pressing "AD" will stop execution and all the KIM monitor functions will be available. Pressing '0' will slow execution rate and pressing any other key (except "ST" & "RS") will speed up operation. "ST" will resume execution at whatever address is on the display, so be sure to hit "PC" to restore the program count to the display, first. Of course, you must be sure your program's use of memory does not conflict with SLOW STEPPER's (also timer use). Terminate your program with a BRK (00) instruction so it will stop when done.

HOW IT WORKS

When an interrupt occurs, the 6502 will complete the current instruction then save the program counter high byte, PC low byte and the processor status (flags) register on the stack in that order. The 6502 then jumps to the location specified by the interrupt vector and begins executing instructions at that address. At the 1C00 (SAVE) KIM monitor NMI entry point the monitor program will save all the processor registers in fixed zero page locations, including pulling and storing those that were pushed onto the stack. Also the current stack pointer is saved in its designated location. All of these stored values may be examined or changed via the monitor program using the KIM keyboard. Normally, when "GO" is pressed, the monitor will load the values in the storage registers into the 6502's registers and appropriate stack locations using the instructions starting at 1DC8 (GOEXEC), jumping into the user program via the RTI instruction at 1DDA commencing at the location displayed on the address portion of KIM's display. If the SST switch is on, KIM's hardware will create an interrupt during the fetch of the first opcode after leaving the monitor program. This single instruction will be executed and with 1C00 as the NMI vector, control is returned to the monitor program. KIM's hardware prevents interrupts from occuring while in the monitor program. To create a slow stepping routine, we must duplicate the SST action on a recurrent basis. Using the SST circuitry is not possible without hardware being added as the existing hardware will interrupt every instruction not in the monitor, and there is no way to automatically return from the monitor. Our solution is to use an interrupt vector pointing to a routine which duplicates the KIM monitor action in storing registers, then waiting out a delay loop. Following this delay, a timer interrupt is set to go off immediately following the time interval it takes to jump to the monitor routine at 1DC8 (GOEXEC) which restores the registers and then executes an RTI instruction. We set the timer for a 40 microsecond delay, which is exactly the correct time for all of this to occur and create an interrupt on the next opcode fetch exactly as the SST hardware would do it. The instruction is executed, the processor does its interrupt thing and jumps to 0119 (NEXT) in SLOW STEPPER which duplicates the KIM monitor "SAVE" procedures, completing the cycle. During the delay loop the program looks for an input from the keyboard, escaping to regular monitor functions in response to an "AD" key, slowing the step rate in response to a "0" key or speeding up in response to any other key (except "ST" or "RS"). The rate of stepping is determined by the number of loops it takes to decrement "WAIT" to zero from the "SPEED" value, which varies from its initial value of FF (slowest) to 01 (fastest). This value is changed by either dividing or multiplying by 2 in response to key entry.

```
0100 A9 19       START   LDA #19      Set NMI vectors to halt address
0102 8D FA 17            STA NMIL
0105 A9 01               LDA #01
0107 8D FB 17            STA NMIH
010A A2 00               LDX #00      Initialize port to enable timer
010C 8E 03 17            STX PBDD     interrupt
010F 86 ED               STX HOLD     Start delay
0111 CA                  DEX
0112 86 EC               STX SPEED    Stepping speed count
0114 C6 ED       HALT    DEC HOLD     Negative is start/restart mode
0116 4C 16 1C            JMP NOSAV    To KIM without saving registers
0119 2C 04 17    NEXT    BIT RDCLK    Timer interrupt, clear timer latch
011C 24 ED               BIT HOLD     Test for start or step mode
011E 10 06               BPL STEP     Positive is step
0120 68 68 68            PLA PLA PLA  Negative is start, adjust stack
0123 38                  SEC
0124 B0 16               BCS NOSTEP   Unconditional branch
0126 85 F3       STEP    STA ACC      Save all the registers just like
0128 68                  PLA          KIM monitor
0129 85 F1               STA PREG
012B 68                  PLA
012C 85 EF               STA PCL
012E 85 FA               STA POINTL
0130 68                  PLA
0131 85 F0               STA PCH
0133 85 FB               STA POINTH
0135 84 F4               STY YREG
0137 86 F5               STX XREG
0139 BA                  TSX
013A 86 F2               STX SPUSER
013C A5 EC       NOSTEP  LDA SPEED    Transfer value to counter for
013E 85 EE               STA WAIT     controlling stepping rate
0140 A5 ED               LDA HOLD     Test mode
0142 10 07               BPL NODEC    Skip delay if step
0144 20 19 1F    XDELAY  JSR SCAND    Display and delay if start
0147 C6 ED               DEC HOLD
0149 D0 F9               BNE XDELAY   Wait out the count
014B D8          NODEC   CLD          Binary mode for keys
014C 20 19 1F    SPDLP   JSR SCAND    Display address & opcode
014F F0 19               BEQ NOKEY    No key, skip key routine
0151 20 6A 1F    JSR     JSR GETKEY   Key down
0154 C9 10               CMP #10      AD key?
0156 F0 BC               BEQ HALT     Yes, stop stepping & enter Monitor
0158 AA                  TAX          For later test
```

```
0159 A5 EC              LDA SPEED    Value for current stepping rate
015B CA                 DEX          Test for 0 key
015C 30 02              BMI SLOW     Yes, slow step rate
015E 4A 4A              LSR LSR      All other keys, divide value by 4
0160 0A                 ASL          Multiply value by 2
0161 09 01              ORA #01      Minimum value = 01
0163 85 EC              STA SPEED    Store new value (2X or .5X)
0165 20 19 1F   DOWN    JSR SCAND    Wait here until key is released
0168 D0 FB              BNE DOWN
016A C6 EE      NOKEY   DEC WAIT     Loop til countdown done for step time
016C D0 DE              BNE SPDLP
016E A5 F9              LDA INH      Check opcode
0170 F0 A2              BEQ HALT     Stop if it's a BRK (00)

0172 A9 28              LDA #28      otherwise set timer for interrupt
0174 8D 0C 17           STA CLK1T    at next opcode in program
0177 4C C8 1D           JMP GOEXEC   KIM will restore all registers
                                       & return to program
```

letters & comments

John F. Cowan
1150 Polk Ave.
Sunnyvale, Ca 94086

Dear Eric,

I have been reading the KIM User Notes for some time now and enjoying them. Here are some ideas and comments which might be of interest to you and your readers.

I purchased a 4K memory board from Solid State Sales in Cambridge, Ma and was very disappointed with it. I built it without sockets for the memory chips, half of which later proved to be defective. I would discourage anyone from buying this board. If the board is purchased, sockets should definitly be used for all chips.

I have purchased a 16K memory system, motherboard and KIM interface from Katherine Atwood Assoc. in Santa Ana, Ca and am very pleased with it. The 4K memory boards are only $89, assembled, tested and guaranteed, a beautiful idea. I have been running with the four boards for two months now and have yet to drop a bit. The motherboard kit costs $30, but is free if you buy four 4K boards. The KIM interface kit costs $24.50 and plugs right in to the KIM expansion connector. It allows memory addresses to be assigned at will and provides for memory protect. It is also a real bargain. I thought long and hard before purchasing this system rather than a KIMSI and I have not been disappointed. I am most happy with the fact that the system is compact (cards are 4½x7"), the bus is simple and straightforward and the system components are of industrial quality. Numerous other boards are available, including 8K Prom, Prom burner, Analog in, Analog out and Digital out (64 lines using AMI 6820's for $56!). I highly recommend this system to all KIM users.

Last, a few words about my activities. I have just finished implementing a word processor system after interfacing a KIM to a Selectric I/O Writer; a tour de force lasting two and a hlf years. I have liked the KIM since day one and grown to like that strange programmable beast, the 6502. This letter was written and edited with the word processor. I am considering making it available, perhaps via the 6502 Program Exchange.

****

RANDOM COMMENTS ABOUT KIM & SYM from-

Bob Leedom
14069 Stevens Valley Ct.
Glenwood, Md. 21738

Pet peeve -- Having to go through each line of a well-written, very useful "utility program" to determine where the program's variables are located in memory. It has happened more than once that a utility program sneakily altered a location or two of a user program's variables. Moral - Document all memory used by a program!

I was finally able to play the ASTEROID game in First Book of KIM the other day! I circumvented my bouncy keyboard (disastrous in ASTEROID!) with the super-cheap A/D in KUN #4 (p. 9) and a slide-type variable resistor, giving me a neat "spaceship controller". (My three little ones, aged 6, 4, and 4, are now veteran pilots.) But I discovered an awful bug in the program as published. Since only the ADL of the pointer is incremented at $2B3, and since the asteroid field crosses a page boundary, there is a point at which the asteroids you're dodging are made up from the program code itself, and the field is impassable! In other words, the pointer is now pointing into the program, rather than at the stored field pattern. Easy fix (though to be honest, I haven't tried it -- I patched up the code) -- relocate the field entirely in page 3 and change 2CE and 2CF to point there.

Dear fellow programmers, I will if you will: give constants and variables and subroutines Names instead of referring to their location in memory. It makes your program much easier to read, discuss, understand, modify, and/or relocate. JSR $01B2 drives me buggy: JSR DISPLAY makes me think I know what you're doing.

If nobody else has mentioned it, I would like to say many thanks to Timothy Bennett for the index to Volume 1 of KUN -- I use it all the time. I hope that between Eric and Timothy, this will not be the last of the indexes (indices?).

Has anybody else had the unbelievably depressing job of trying to help a friend with a SYM (formerly VIM)? A guy I know invested his bucks in the SYM, thinking he would get all the power of SUPERMON and the dual cassette outputs and the plug-in expansion memory, etc. plus the ability to use all the currently available KIM programs simply by changing a few subroutine addresses according to the list in Appendix E of the SYM manual. Let me quote from paragraph E.2 of this manual: "Many of the routines do not perform identically in the two systems, however, and you should check their operation in Table 9-1 before using them." Folks, I'm here to tell you, it ain't all that simple. One little example: SYM's GETKEY returns ASCII code for the key depressed. That means every KIM program that uses a key from $0 to $F as an index or as a number has to have several extra lines of code patched in for SYM use so that the $30 can be stripped off of 30 through 39, and $36 can be subtracted from the A through F keys. Don't get me wrong, the SYM-1 is a super little machine: but KIM compatibility is not straightforward.

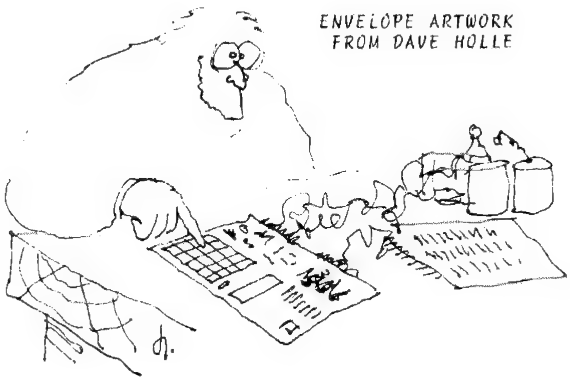
COMMENTS ABOUT SYM

R. Bruce Harvey
52 Spruce Drive
Truro, N. S.
Canada B2N 4X6

"I bought my KIM-1 in August and have really been enjoying it since then. As I teach Physics in grade 11 and 12 here and have had an electronics club in the school we decided this year to purchase a micro computer for the electronics club. I decided to purchase the SYM-1 board and have it now. The advertisements were a bit deceptive to me and I thought that the KIM programs I had prepared, mainly from your notes were going to be easily modified to work with the SYM.

I have had no trouble in getting the programs to load and after altering them according to the conversion chart supplied with the SYM-1 manuals I still have not been able to get the programs to run. I realize that I am new at this, although I have been in amateur radio for a little over seventeen years, and there are others who have probably not encountered my problems. I wonder if you would have any information that would help me to get these programs to run. I had hoped that the programs I have on tape could be used although it now seems that I will have to modify the written programs before loading them into the SYM-1 by hand. The information supplied with the SYM-1 appears to be incomplete with regards to the use of KIM software. In particular the monitor does not seem to be operating. The clock program that was in the last issue of Micro is working well as well as are the various samples in the SYM-1 manual. Any information you could provide me with would be very much appreciated."

(EDITOR'S NOTE - I'm aware of the problems involved in trying to convert KIM programs over to the SYM. At first glance, it seems that new scan routines would have to be written to simulate KIM I/O on the SYM, but that's just my first observation (somewhat hurried at that). Maybe one of our readers has already solved this problem. HOW ABOUT IT ??? HERE'S ANOTHER ARTICLE IDEA !!!!)



ENVELOPE ARTWORK
FROM DAVE HOLLE

software library

Here's a useful addition to your software library. Now you don't need to spend $50 for that TI programmers calculator. Can you convert this to run on a terminal ???.........

MULTI-MODE ADDER

Jim Butterfield
Toronto

This program adds and subtracts, in either decimal or hexadecimal.. and will convert between the two number systems. So if you hit keys DA (for set Decimal), 123, AD (for set Hexadecimal), you'll see the hex equivalent of 123 which is 7B. Hit DA again and you're back to 123 decimal.

Negative numbers are held in complement form; so FFFFE9 is equal to minus 17 hexadecimal, or 999972 would be minus 28 decimal. You can reverse the sign on a number by hitting GO (for Clear) followed by PC (for subtract).

Meaning of the KIM keys is as follows:

```
     GO = Clear ..set the total to 000000.
     AD = Hexadecimal mode .. convert display to
          Hexadecimal
     DA = Decimal mode .. convert display to Decimal
     PC = Subtract .. subtract last number entered
          from total
      + = Add .. add last number entered to total
```

The Add and Subtract keys "chain" ..so you can add a number repeatedly if you wish .. or if you have added an incorrect number in error, pressing Subtract (PC) will subtract it again.

You should always begin by pressing GO (Clear) followed by AD or Hexadecimal or DA for Decimal .. otherwise you won't know what mode you're in. The program does not warn of overflow, so be careful if you're dealing with large numbers.

All numbers are held in 24-bit binary in the computer .. they are translated to Hexadecimal or Decimal for the display. The program for translating this is quite compact, and may be found at addresses 0243 to 0259. For converting Decimal input to binary, a much longer program is located at 0280 to 02C2.

Example: a program starts at hex 0200 and goes to 0352. How many decimal locations does it occupy?

```
     GO (clear) AD (Hex) 0352 + 0200 PC (-);
          display shows 0152;
     DA (Decimal); display shows 338
     1 + (since the numbers are inclusive); display
          shows 339 locations.
```

Note that the program uses Polish notation, i.e., enter the number first, then the add or subtract code.

The Clear (GO) key sets the total to 000000, and transfers the previous total to the "chaining" register. Thus, you can restore the total by hitting +, double the previous total by hitting + twice, or complement the previous total by hitting Subtract (PC).

```
0200 D8         START   CLD
0201 20 1F 1F           JSR SCANDS      light display
0204 20 6A 1F           JSR GETKEY
0207 C5 D0              CMP LAST        same as last key?
0209 F0 F5              BEQ START       yes, nothing to do
020B 85 D0              STA LAST        save new key ID
020D C9 13              CMP #$13        GO=clear?
020F D0 52              BNE NOGO
0211 A2 02              LDX #2
0213 A0 00              LDY #0
0215 B5 D4      MLP     LDA TOT,X       move total to..
0217 95 D7              STA INC,X       ..input area
0219 94 D4              STY TOT,X       and zero total
021B CA                 DEX
021C 10 F7              BPL MLP
021E A9 00      HEXDC   LDA #0          convert TOT (hex) to display
0220 85 D1              STA DISP        display-total flag
0222 A2 02              LDX #2          clear display..
0224 B4 D4      DLP     LDY TOT,X       ..and conv TOT to WORK
0226 94 DA              STY WORK,X
0228 95 F9              STA INH         zero display area
022A CA                 DEX
022B 10 F7              BPL DLP
022D A5 D6              LDA TOT+2       test sign
022F 85 D3              STA SIGN
0231 10 0C              BPL WUP
0233 38                 SEC             invert WORK
0234 A2 FD              LDX #$FD          ..3 digits
0236 A9 00      WLP     LDA #0
0238 F5 DD              SBC WORK+3,X
023A 95 DD              STA WORK+3,X
023C E8                 INX
023D 30 F7              BMI WLP
023F A5 D2      WUP     LDA MODE
0241 F0 01              BEQ HEXOUT      hex display
0243 F8                 SED             decimal display
0244 A0 17      HEXOUT  LDY #23         24 bits to translate
0246 06 DA      SWING   ASL WORK
0248 26 DB              ROL WORK+1      get most significant bit
024A 26 DC              ROL WORK+2      ..into carry
024C A2 FD              LDX #$FD
024E B5 FC      HUFF    LDA POINTH+1,X
0250 75 FC              ADC POINTH+1,X
0252 95 FC              STA POINTH+1,X
0254 E8                 INX
0255 30 F7              BMI HUFF
0257 88                 DEY
0258 10 EC              BPL SWING
025A A5 D3              LDA SIGN
025C 10 A2              BPL START
025E 20 3F 03           JSR FLIP
0261 10 9D              BPL START
0263 C9 10      NOGO    CMP #$10        Operation or Numeric?
0265 90 6E              BCC NUML
0267 C9 15              CMP #$15        No key?
0269 F0 95              BEQ START
026B A4 D1              LDY DISP        already converted input?
026D F0 71              BEQ OK
026F A5 D2              LDA MODE
0271 F0 64              BEQ HEXIN       hex mode input?
0273 A5 FB              LDA POINTH      decimal sign test
0275 69 30              ADC #$30
0277 85 D3              STA SIGN
0279 10 05              BPL POZZ        positive number entered?
027B F8                 SED
027C 20 3F 03           JSR FLIP        invert negative input
027F D8                 CLD
0280 A9 00      POZZ    LDA #0          Dec-Hex conversion starts here
0282 A2 02              LDX #2
0284 95 D7      CLP     STA INC,X       Clear Hex input area
0286 CA                 DEX
0287 10 FB              BPL CLP
0289 A0 05              LDY #5          Six decimal digits to come
028B A9 00      DIG     LDA #0
028D 20 32 03           JSR GETD        Get highest digit
0290 85 DE              STA DIGIT
0292 20 4C 03           JSR ROLIN       Multiply INC by two...
0295 A2 02              LDX #2          ..and copy..
0297 B5 D7      ILP     LDA INC,X       ..INCx2 into..
0299 95 DA              STA WORK,X      ..WORK;
029B CA                 DEX
029C 10 F9              BPL ILP
029E 20 4C 03           JSR ROLIN       Multiply INC by four,
02A1 20 4C 03           JSR ROLIN       ..giving INCx8
02A4 A2 FD              LDX #$FD        ..then add INCx2 from WORK
02A6 B5 DA      TLP     LDA INC+3,X          ..giving..
02A8 75 DD              ADC WORK+3,X         ..INCx10
02AA 95 DA              STA INC+3,X
02AC E8                 INX
02AD 30 F7              BMI TLP
02AF A2 FE              LDX #$FE
02B1 A5 D7              LDA INC         Now add the new digit..
02B3 65 DE              ADC DIGIT        ..to INCx10
02B5 85 D7              STA INC
02B7 B5 DA      ZLP     LDA INC+3,X     ..propogating any carry
02B9 69 00              ADC #0             ..into the higher digits
02BB 95 DA              STA INC+3,X
02BD E8                 INX
02BE 30 F7              BMI ZLP
02C0 88                 DEY             on to next decimal digit
02C1 10 C8              BPL DIG              ..if any
02C3 A5 D3              LDA SIGN
02C5 10 19              BPL OK           SIGN OK?
02C7 38                 SEC
02C8 A2 FD              LDX #$FD         no, re-invert
02CA A9 00      FLAP    LDA #0
02CC F5 DA              SBC INC+3,X
02CE 95 DA              STA INC+3,X
02D0 E8                 INX
02D1 30 F7              BMI FLAP
02D3 10 0B              BPL OK
02D5 90 41      NUML    BCC NUM
02D7 A2 02      HEXIN   LDX #2           Hex input:
02D9 B5 F9      HLP     LDA INH,X         copy to INC
02DB 95 D7              STA INC,X
02DD CA                 DEX
02DE 10 F9              BPL HLP
```

```
02E0 A2 FD     OK     LDX #$FD
02E2 A5 D0            LDA LAST
02E4 29 0F            AND #$0F
02E6 C9 02            CMP #2             AD (Hex) or DA (Dec) key?
02E8 B0 10            BCS ACT            no .. must be PC or +
02EA 85 D2            STA MODE           set mode to Hex or Dec
02EC A5 D1            LDA DISP           total or entry?
02EE F0 07            BEQ EXIT           total, do nothing
02F0 B5 DA     TLP    LDA INC+3,X        entry, move to total
02F2 95 D7            STA TOT+3,X
02F4 E8               INX
02F5 30 F9            BMI TLP
02F7 4C 1E 02 EXIT    JMP HEXDC
02FA D0 0C     ACT    BNE NADD           not + .. must be PC (-)
02FC 18               CLC
02FD B5 D7     ALP    LDA TOT+3,X
02FF 75 DA            ADC INC+3,X        add (binary)
0301 95 D7            STA TOT+3,X
0303 E8               INX
0304 30 F7            BMI ALP
0306 10 EF            BPL EXIT
0308 C9 04     NADD   CMP #4
030A D0 EB            BNE EXIT
030C 38               SEC
030D B5 D7     SLP    LDA TOT+3,X
030F F5 DA            SBC INC+3,X        subtract (binary)
0311 95 D7            STA TOT+3,X
0313 E8               INX
0314 30 F7            BMI SLP
0316 10 DF            BPL EXIT
0318 A4 D1     NUM    LDY DISP           first digit?
031A D0 0A            BNE ROLL           no, shift it in
031C 84 FA            STY POINTL         zero into display..
031E 84 FB            STY POINTH           ..except..
0320 85 F9            STA INH                ..new digit
0322 C6 D1            DEC DISP
0324 30 09            BMI OUT            unconditional exit
0326 20 32 03 ROLL    JSR GETD         make space for new digit
0329 A5 D0            LDA LAST         ..and insert it
032B 05 F9            ORA INH
032D 85 F9            STA INH
032F 4C 00 02  OUT    JMP START
               ;  subroutines
0332 A2 03     GETD   LDX #3           move four bits
0334 06 F9     GLP    ASL INH          through display
0336 26 FA            ROL POINTL
0338 26 FB            ROL POINTH
033A 2A               ROL A
033B CA               DEX
033C 10 F6            BPL GLP
033E 60               RTS
               ;
033F 38        FLIP   SEC                  complement
0340 A2 FD            LDX #$FD               the
0342 A9 00     FLP    LDA #0                  contents
0344 F5 FC            SBC POINTH+1,X           of
0346 95 FC            STA POINTH+1,X            the
0348 E8               INX                        display
0349 30 F7            BMI FLP                     area
034B 60               RTS
               ;
034C 06 D7     ROLIN  ASL INC            multiply
034E 26 D8            ROL INC+1            INC
0350 26 D9            ROL INC+2             by
0352 60               RTS                     2

0353 end
```

A PSEUDORANDOM NUMBER GENERATOR

H. T. Gordon
641 Paloma Avenue
Oakland, Ca 94610

This is the 6502 code of my 8080-coded program that will be published in BYTE. I am copyrighting the 6502 version with this "free-diffusion" clause: Any and all uses are authorized if (and only if) all software associated with the whole or any part of the following coding is declared to be equally available for unrestricted use by everyone.

Subroutine MIXRND will generate 65K binary 8-bit numbers before repeating. It uses 3 zero-page locations (in this case C1, C2, and C3, but any other locations will do and they need not be in sequence), that need no initialization. The subroutine is fully relocatable.

```
0110  E6 C3    MIXRND INC RND+2 (one of 256 se-
                                 quences)
  12  D0 02           BNE SEQUEN (same sequence)
  14  E6 C2           INC RND+1 (increment addend)
0116  A5 C1    SEQUEN LDA RND   (load seed)
  18  0A              ASL A     (X 2)
  19  0A              ASL A     (X 4)
  1A  18              CLC
  1B  65 C1           ADC RND   (X 5)
  1D  18              CLC
  1E  69 2B           ADC #$2B  (add for next
                                 seed, $6B, $AB,
                                 or $EB also work)
0120  85 C1           STA RND   (store next seed)
  22  18              CLC
  23  65 C2           ADC RND+1 (add sequence ad-
                                 dend)
  25  24 C1           BIT RND   (seed bit 7=N, bit
                                 6=V)
  27  30 03           BMI TESTV (N=1)
  29  50 05           BVC EXIT  (V=0)
  2B  B8              CLV       (reset V)
  2C  70 02    TESTV  BVS EXIT  (bypass comple-
                                 menting)
  2E  49 FF           EOR #$FF  (complement output
                                 in A)
0130  60       EXIT   RTS       (pseudorandom # is
                                 in A)
```

Subroutine SELBIT can be used to screen the MIXRND output and yield sequences of non-binary numbers. E.g., if RND+3 is pre-set to $A0 and RND+4 to $0A, SELBIT will exit with the carry clear if MIXRND has output one of the 100 BCD numbers from 00 to 99. If the carry is set, MIXRND can be repeatedly called until it outputs a BCD.

```
0131  C5 C4    SELBIT CMP RND+3 (compare hi nyb-
                                 ble)
  33  B0 06           BCS SELOUT(reject    or =)
  35  48              PHA       (save # in stack)
  36  25 0F           AND #$0F  (retain lo nybble)
  38  C5 C5           CMP RND+4 (compare)
  3A  68              PLA       (restore # in A)
013B  60              RTS       (accept if carry
                                 clear)
```

By using $D0 and $0D, one would get a pair of "tridecimal" numbers to simulate playing cards. The "suit" could be established by using the 2 low-order bits of the output for the high-nybble card, and of RND for the low-nybble card, with additional logic to eliminate duplications.

ASCII DUMP PROGRAM

Jim Zuber
20224 Cohasset #16
Canoga Park, Ca 91306

This program is written for the KIM-1 to SWTPC PR-40 printer interface I described in issue #11 of the User Notes. This program will dump ASCII data from memory, decoding carriage returns (HEX 0D) and a special end of data character that can be defined by the user. I am using this program to print mailing lists and have used this program as a subroutine in larger programs. (Just change location 00D0 to 60)

To use the program jsut do the following:

1. Store the starting address of the ASCII data in 000A and 000B (low order first)

2. Set the last character in the ASCII string to "@" (HEX 40)---Note:if you want to use a different character for the end of data marker set location 008F to the HEX equivalent of the ASCII character you want to use.

3. Start the program at 0080 and you will get an ASCII dump.

```
        ----ASCII DUMP----

0080   A9 FF 8D 01 17 A9 01 8D 03 17 A0
008B   00 B1 0A C9 40 F0 3E C9 0D F0 1F
0096   8D 00 17 A9 01 8D 02 17 CE 02 17
00A1   EE 02 17 18 A5 0A 69 01 85 0A A5
00AC   0B 69 00 85 0B A9 00 F0 D7 A9 0D
00B7   8D 00 17 A9 01 8D 02 17 CE 02 17
00C2   EE 02 17 AD 02 17 29 02 F0 F9 A9
00CD   00 F0 D4 4C 4F 1C
```

KEYBOARD DEBOUNCE ROUTINE

Thomas J. Rubens
851 California St.
San Francisco, Ca 94108

The following code performs seeming miracles on noisy keyboards. The standard implementation of the KIM-1 monitor code wrongly assumes that inexpensive keyboards are not inherently noisy.

The code was inspired by Allen Anway's Program Branch from "Notes" 9 & 10.

CTR is any convenient page zero address.

```
A0 05        SCN0  LDY  #05  Set up safety net
84 EE              STY  CTR
20 19 1F     SCN1  JSR SCAND
D0 F7              BNE SCN0  Wait for key release
C6 EE              DEC  CTR  Make sure it
D0 F7              BNE SCN1  Wasn't noise
20 19 1F     SCN2  JSR SCAND New key pressed?
F0 FB              BEQ SCN2  No
20 19 1F           JSR SCAND Yes - check again
F0 F6              BEQ SCN2  No
20 6A 1F           JSR GETKEY Yes-get key immage
```

STAR WARS BATTLE

Jim Zuber
20224 Cohasset #16
Canoga Park, Ca 91306

Want some wild sound effects for your KIM? I have combined Ron Kushniers space wars phaser sound program with Jim Butterfield's random number generation to create sound effects from an entire battle scene out of Star Wars!! Interesting variations can be obtained by changing the mask byte for the random number. Location 0247 controls the number of repeats and 0254 controls the time of the phaser pulse. The program starts at 0241 and the sound output is at PA-0.

```
0200  A0 03 A9 00 85 EE A9 11 8D 06 17
020B  A9 01 8D 01 17 EE 00 17 A6 EE CA
0216  D0 FD 2C 07 17 10 F3 E6 EE A5 EE
0221  C9 FF F0 02 D0 DF 88 F0 02 D0 DA
022C  60 D8 38 A5 13 65 16 65 17 85 12
0237  A2 04 B5 12 95 13 CA 10 F9 60 20
0242  2D 02 A5 12 29 03 8D 01 02 EE 01
024D  02 20 2D 02 A5 12 29 13 8D 07 02
0258  EE 07 02 20 00 02 4C 41 02
```

SOUND EFFECTS PROGRAM

Bob Carlson WA6QXX

I have been using KIM'S cassette audio output port (SBD at $1742) for outputting music and modern programs. No external hardware aside from a cassette player and an earphone or speaker are required. Simply plug the earphone or speaker into the monitor jack and push down the record button and high fidelity output will result. On my cassette player the tape doesn't even have to be moving. I think this is the simplest interface for audio experimenting yet.

I came up with the following program which makes quite an interesting noise - similar to a police siren or an alarm, using the above mentioned output method.

```
0100 A2 FF      START LDX #$FF Send 1's to
0102 8E 42 17         STX SBD  Output Port
0105 A6 00            LDX #$00 Load Freq Parameter
0107 CA         LOOP1 DEX      Wait Loop For
0108 D0 FD            BNE LOOP1 Waveform High Time
010A A2 00            LDX #$00 Send 0's To
010C 8E 42 17         STX SBD  Output Port
010F A6 00            LDX #$00 Load Freq Parameter
0111 CA         LOOP2 DEX      Wait Loop For
0112 D0 FD            BNE      Waveform Low Time
0114 C6 00            DEC      DEC Freq Once Each
                                              Loop
0116 4C 00 01         JMP      Start
```

MELODIES FOR THE MUSIC BOX

Douglas Lyon
125 Stratton Rd.
New Rochelle, N.Y. 10804

Everyone who owns a KIM should also own The First Book of KIM. If they don't, they should get one, it's worth it. On page 88 of the book you will find Jim Butterfield's Music Box program. Load it. Mr. Butterfield wrote this program real well but he didn't include enough music for us music buffs! So load the following into KIM and you should get 1. Pop Goes the Weasel 2. Happy Birthday 3. London Bridges Falling Down 4. Ten Little Indians and 5. a short version of the Marine Hymn. The second hex dump is a more jazzed up version of the Marine's Battle Hymn I'm sure you'll enjoy it.

```
0000 FB 30 FE 00 FD 01 FC 02 C0 40 B9 39 32 32 32 C0
0010 C0 40 B9 2F A9 C0 40 B9 39 32 29 32 C0 AF 80 B9
0020 2F A9 C0 80 80 80 FB 22 FC 02 FD 01 FE 56 56 CD
0030 D6 C0 C8 56 56 CD D6 D9 C0 56 56 A9 B2 C0 E8 CD
0040 2F 2F B2 C0 B9 80 80 80 80 FB 50 FC 02 FD 02 FE
0050 FF 5D 32 AF 39 32 AF 39 32 2F 29 AF B9 2F 29 2F
0060 32 39 32 AF 40 39 B2 39 32 AF 2F 29 2F 32 39 32
0070 AF CO AF 39 C8 00 80 80 80 80 80 80 80 80 FB 20
0080 FC 02 FD 01 FE FF C8 48 48 C8 48 48 B9 2F 2F 39
0090 40 C8 C0 40 40 C0 40 40 CD 40 40 4D 56 E2 C8 48
00A0 48 C8 48 48 B9 2F 39 4D C8 AF 32 32 39 39 C0 C8
00B0 80 80 80 80 FB 30 FC 02 FD 03 FE FF 62 48 C0 C0
00C0 CO C0 C0 2F C0 4D 48 C0 C0 48 D6 E2 E2 62 48 C0
00D0 C0 C0 C0 C0 2F C0 4D 48 C0 C0 48 D6 E2 E2 FF 00
```

Jazzed Up Marine's Hymn

```
0000 FB 30 FC 02 FD 03 FE FE 62 48 C0 C0 C0 C0 C0 2F
0010 C0 4D 48 C0 C0 48 D6 E2 E2 62 48 C0 C0 C0 C0 C0
0020 2F C0 4D D6 C0 C0 48 D6 62 62 AF 32 B9 C8 B9 AF
0030 C0 4D C0 DF 32 B9 C8 39 AF C0 62 4D C0 C0 C0 C0
0040 C0 2F C0 4D 98 C0 C0 40 AB AF 80 80 80 80 80 FF
0050 00.
```

"DO LOOPS" FOR KIM

Dave Skillman
9514 48th Ave.
College Park, Md.

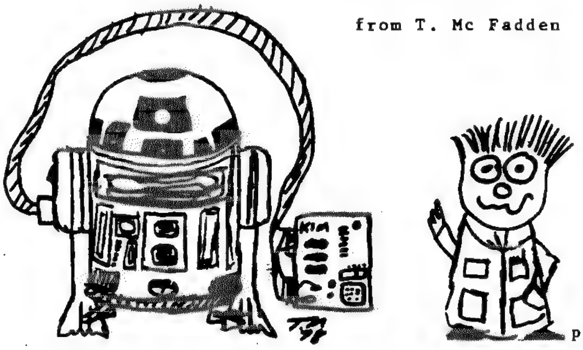
There is often a need to repeat a section of code a given number of times. The following instructions show one way to perform the "do loop" function by executing a block of code N times.

```
       LDA   #00     load zero
       STA    I      ready do loop variable
LOOP   INC    I      increment loop variable
       LDA    N      get loop iteration limit
       CMP    I      compare to present value
       BCC   OUT     branch away if I is greater
                                         than N

       Block of instructions
       to be executed N times

       JMP   LOOP    loop back until done
OUT    BRK           stop if job is done
```

In complex programming situations it is often clarifying to code in a high level language first, and to translate that to assembly code as a second step.

MORE ENVELOPE ART

from T. Mc Fadden



interface

CAMERA SPEED TESTER

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

While it would probably not be a terrific idea to buy a KIM-1 just for use of the display, some projects can be carried out for a lot less money once you own one.

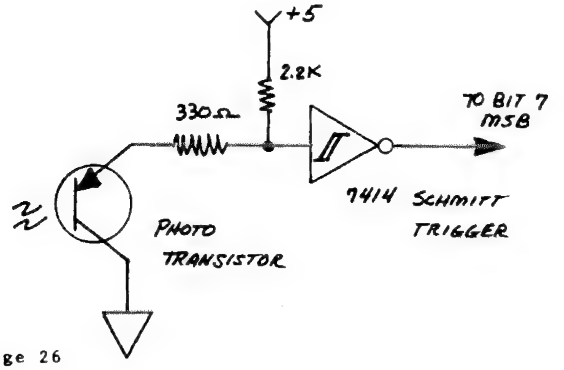
A good example is the camera speed tester that appeared in Popular Electronics. When you own a KIM-1, construction of this device is so trivial that it can be an instant breeze. You can do all the timing in software and not build the display, using KIM for that.

The only piece of hardware you have to have is the photocell and a guarantee that the input will be logically on or off. This can be done with a photo transister and a 7414 Schmidt trigger invertor. I bought a small photo transister from Radio Shack (276-130), wired a 330 ohm resister in line, put glue on the leads for insulation and, after reaming out the hole a bit, slid it into the back end of an empty Bic pen. I used some black ink to darken the plastic. This is my general purpose phototransister tester. For the camera tester, I drilled a hole in a piece of wood the right size to take the pen and to fit where the film goes on the back of the camera. The circuit below shows the 7414 in use, the output simply being taken to one of the ports on KIM.

Basically the programming consists of loading the timer when the shutter is openned and getting a value when the shutter closes. Because it is a countdown timer, the recovered value must be subtracted from the original. The program shown here will (in theory) measure from 1/1000 to ½ second. It outputs a hexadecimal value. Additions program (which I don't care to do, since I don't care aobut my camera speed that much) would go for specific accuracy and conversion to decimal.

After you have positioned the camera, with the photocell replacing the film, under a fairly bright light, push the GO button. The program sits in a loop, waiting for a change in the input value. Using bit 7 of the input port, as soon as the light hits the transister, we load the time and go into a new loop. When the light goes out, we unload the timer, subtract, store in F9 and loop through SCANDS. Hitting the reset button will automatically place you at the start of the program again. If the display does not come on, test the photocell by covering with your finger; it may not be getting enough light to switch. (Note that it is much more sensitive to incandescent than florescent.)

```
  LDX FF
A LDA PORT
  BPL A     LOOP UNTIL MSBIT=1/NEGATIVE
  STX TIMERX1024      (SHUTTER OPEN)
B LDA PORT
  BMI B     LOOP UNTIL MSBIT=0
  LDA TIME            (SHUTTER CLOSED)
  STAZ F9   TEMPORARY STORE
  TXA
  SBCZ F9   SUBTRACT END VALUE FROM START
  STAZ F9
C JSR SCANDS   LOOP IN DISPLAY
  CLC
  BCC C
```



LOW COST MODEM POSSIBILITY

ERIC

Modems are usually expensive, not readily available, and could be a real pain to get functioning correctly.

What we 6502 users need is a software approach to this problem instead of hardware methods. "Doing it in software" makes much more sense for hobbyists who have more time than money and want to learn the ins-and-outs of computing.

Well, thanks to some TRS-80 users who seem to share our views on the software approach, we now have that alternative.

It's the "MICKEY MODEM" and was published in the November '78 issue of Kilobaud (pg. 52).

The "MICKEY MODEM" consists of only two I.C. amplifiers, a VU meter, and a telephone line isolation transformer. (Not to mention some assorted resistors, capacitors, switches etc).

This low-cost circuit contains none of the usual modem thingamajigs such as frequency generators, receiving filters, originate-answer mode switching etc., but interfaces directly to the phone line (instead of the usual acoustic coupling) and relies on the computer to generate the proper tones to transmit and decode the tones when receiving. The utmost in simplicity.

The only addition I would make would be a schmitt-trigger or comparator on the output of the interface to clean up the waveform and make it easier to decode.

The driving software is completely open at this point. Initially I am looking at the Kansas City format (1200 Hz / 2400 Hz tones, 300 baud) since it is fairly straight forward to encode and decode and the high tone is well within the telephones 3000 Hz upper limit.

Perhaps I'll have one of these in operation by the next issue. Any comments and/or ideas would be greatly appreciated. Let's hear what you think of this idea.

RPN CALCULATOR CHIP INTERFACE

ERIC

Another interface design for the National MM57109 has been published, this time in Byte Magazine, August '78 (pg. 64).

This interface looks like a perfect match to a 6520 or 6522 I/O chip.

The software driver presented was for the 8080.

I will be installing one of these calc chips on my 6522 I/O board (see elsewhere in this issue) using this interface.

This calculator interface would make an excellent addition to Tiny Basic or Focal since they do not have built in trig functions.

Anyone up for the job???

POWER-ON RESET

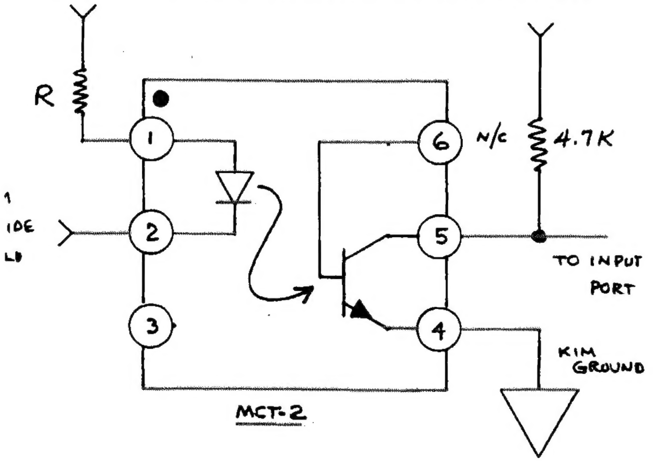
George W. Hawkins, NY

A very simple power-on reset can be added to KIM by connecting a .68uf tantalum capacitor between the bottom end of resistor R4 (+), and the bottom end of resistor R13 (-). This was the smallest value that would work for me. The capacitor is connected across the RS key when connected as described above. See page B-1 to find the resistors, which are to the left of the keyboard.

THE OUTSIDE WORLD CONNECTION

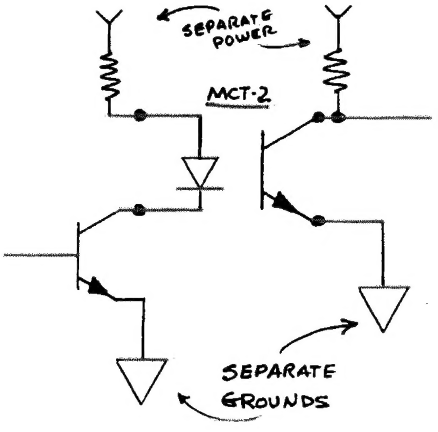
ERIC

When connecting KIM to outside world systems that have their own power supplies - it makes good sense to isolate the two systems and avoid ground loops and other problems. I have been using a MONSANTO MCT-2 optoisolator to perform this task but other opto-isolators may work as well.



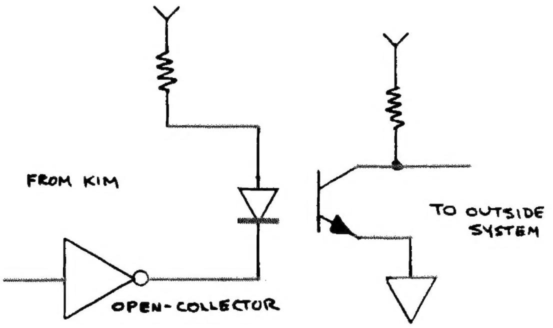
+V can be any reasonable positive dc voltage with R adjusted to allow about 20 ma (no more than 50 ma) through the diode when its cathode is grounded.

An open-collector transistor is a convenient device to trigger the LED.



This circuit has been used to successfully interface an active-filter RTTY (Radio Teletype) demodulator to KIM. The demodulator has an open-collector transistor output to connect to a 5-level teletype with a 60 ma. loop so hooking it to the opto-isolator was a breeze.

The circuit may also be used in the output mode but an open-collector gate should be used to drive the opto-isolator since the KIM output port can't sink the required 20 ma.



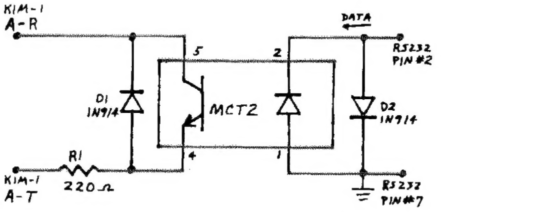
As you can see, opto-isolators are simple to use and handy to have around.

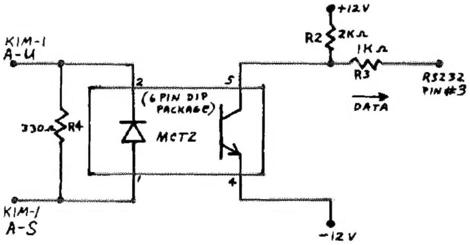
Next issue, we'll discuss the 555 IC timer and see how we can put it to work.

MORE ON THE OPTO-ISOLATOR

Dwight D. Egbert
302 W. 109, #4
NYC, NY 10025

The following KIM-1 TTY to RS232 converter circuit has proven to be very reliable, small, and easy to make. I have used it and KIM-1 with a DecWriter LA-36 (110 & 300 baud), a Tektronix 4012 (110 through 2400 baud), and a Processor Technology 3P+S I/O board (110 & 300 baud). The 3P+S uses the MC1488 and MC1489 RS232 interface IC's which are common to many devices. This converter should work with any RS232 device you have.





NOTES:

1. If you do not have +/- 12 volts on your KIM-1 then your RS232 device should have +/- voltage available. I would expect this circuit to operate properly with anything from about +/- 5 to +/- 15 volts although you might have to adjust R2 and R3.

2. I have not tried the following but it should work if you only have +5 and +12 volts and if your KIM-1 ground is not tied to the AC line voltage ground. IF YOU TRY THIS BE CAUTIOUS! Connect +12 volts as shown. Connect +5 volts to the RS232 ground (pin #7). Connect KIM-1 ground to the point shown for RS232 -12 volts. This makes the RS232 output work at +7 and -5 volts relative.

3. Alternatively, circuits are given in the two following references for conversion circuits which operate from only +5 volts. I have not tried them and don't know how reliable they are. With only +5 volts they cannot meet RS232 specs. even though they might work with some devices.

BYTE, May 1076, "A DAte With KIM", page 10
EDN, June 5, 1977, "Constructing A Low-Cost Terminal Interface", page 205

4. Even though I used Monsanto MCT2 opto-isolators the following all have similar specs. and should work equally well; MCT12, MCT26, MCT6 dual, 4N25, 4N26, 4N27, 4N28, 4N35, and 4N37. Also, Darlington types like MCA230, MCA231, MCA255, 4N29, 4N30, 4N31, 4N32, and 4N33 will work just fine. Radio-Shack offers a grab-bag of opto-isolators and International Electronics Unlimited advertises MCT2's for 70¢ in June 1977 RADIO ELECTRONICS. I have recently become a fan of opto-isolators and recommend you experiment with them. They are great for practically any computer related conversion including AC switching when used with SCR'S.

5. RS232 pin assignments as shown are proper if you want to plug KIM-1 directly into a terminal. Alternatively, if you want to plug KIM-1 into a modem you should reverse the connections to pins 2 and 3.

new products

VIDEO DRIVER PACKAGE

Forethought Products (87070 Dukhobar Rd., Eugene, Or 97402) has announced immediate availability of the 6502 Video Driver Routine (VDR). This software provides the necessary software support for 64x16 random access video display boards (such as Polymorphics VTI, Kent Moore, Solid State Music etc) on systems using the 6502 CPU.

According to the literature, this software allows for complete cursor control, scrolling speed, line & page control, printer control and "partitioning" of the screen into protected areas.

The package includes a 12 page manual with full source listing, and KIM compatible cassette. Two versions are available - one residing at $0200 and the other at $DD00. Both are ROMable and occupy ½K of memory. Price is $9.50. For more info, contact Forethought.

PRICE DECREASE ON JOLT BOARDS

Synertek Systems announced a drop in price on their CP110 Super Jolt CPU boards from $375 to $195.

This 4.5"x7" board contains a 6502, 1K RAM, a 6530 (which contains the ROM monitor, a timer and I/O) and a Xtal clock. The monitor program is identical to TIM (from MOS). Communication is handled through a serial port.

Get more info from Synertek Systems, 150 S. Wolfe Rd., Sunnyvale, Ca 94086, phone 408-988-5682

AN 8080 SIMULATOR FOR THE 6502

Dann McCreary is pleased to announce his 8080 Simulator for the 6502. It joins it's predecessor, the 1802 Simulator. Available now in a KIM-1 version, the 8080 Simulator executes the entire 8080 instruction set. All internal 8080 registers are maintained ready for convenient examination or modification of their contents. In it's minimum configuration on the KIM-1, the 8080 Simulator supports register single-step, program counter single-step and run modes. It also offers an input and an output port, breakpoint operation, and rejection of illegal op-codes.

The 8080 Simulator runs in less than 1K of memory, leaving up to 224 bytes of 8080 programming space on an unexpanded KIM-1. The simulator may be relocated in ROM and can be adapted to other 6502 based systems.

Well suited to all but time-sensitive applications, the 8080 Simulator may be used to assist in the design and testing of 8080 software, used as a training aid or used for running most 8080 application software.

A complete 8080 Simulator package is now available for the KIM-1. It consists of a KIM-1 format cassette tape, a user manual and complete, commented assembly level source/object listing. Priced at $18.00 + $1.50 postage & handling, it may be ordered from: Dann McCreary
Box 16435
San Diego, Ca 92116

Both 8080 and 1802 Simulators purchased at the same time (on the same cassette) are specially priced at $25.00 + $2.00 postage & handling.

California residents please add 6% sales tax.

EPROM PROGRAMMER

Optimal Technology, Blue Wood 127, Earlysville, Va announces the EP-2A-79, EPROM Programmer. Software for programming and verifying programming is available for the 6800, 8080, Z-80, 8085, 6502 (KIM-1), F-8, 1802, and 2650 based microcomputers. Packaged in a sloping panel aluminum case, the unit connects to microcomputer with a 14 pin ribbon cable thru 1½ I/O ports. Software, supplied as a listing, requires approximately 256 bytes of RAM and includes instructions on how to relocate. Personality modules which plug into the front panel-mounted socket, are available for programming the 2708, 2716, TMS 2716, 2732, TMS 2708 and TMS 2532 EPROMS. Power requirements are 115 VAC 50/60 HZ at 15 watts. The EP-2A-79 is priced at $145.00 which includes 1 set of software. Personality Modules are priced at $15.00 except the Personality Modules for the 2732 and TMS 2532 which are $25.00. Available from stock.

EDITORIAL (continued from inside front cover).

For example, their full-size floppy disc system was up and running for almost a year before it was announced to the world. I had the pleasure of using one of these systems for several months while I was still with MOS. This was still months before anyone ever heard of HDE.

This professional and responsible attitude on the part of HDE should be applauded and encouraged.

It was this attitude and the quality of their products which led me to purchase an HDE disc system and memory boards for my system. I will review these products in an upcoming issue.

I can remember, a while back, wondering what I was going to do with this micro-computer now that I had it running and could successfully add two numbers together.

Thats a laugh! Now I wish I had more time to do all the neat things that need doing. There's those mods to Focal, touch-tone software for that now music board, an enclosure for those new discs etc, etc, etc. The list is almost endless. I guess that's the fate of the computer hobbyist.

ERIC

NOTES FROM JOHNSON COMPUTER..................a paid ad

P.O. BOX 523 MEDINA, OHIO 44256 (216) 725-4560

MICROSOFT BASIC for KIM is now available in a PROMable version which stores at 2000-3FFF. This is catalog KB-9P furnished on cassette or paper tape for $99.00, stock. The PROM version does not include SIN, COS, TAN or ATN. Note that it stores on an 8K PROM board such as our KM-8KRO board which plugs directly into a KIM-4 motherboard. The workspace in KB-9P begins at 4000. Most of you have the KB-9 version, which runs out of RAM located at 2000-4260, and does provide the trig functions. Both are a full ANSI BASIC and both provide 9 digit accuracy with a floating point math package.

OHIO SCIENTIFIC's new low priced computer, model C1P, breaks all records for performance/price by offering, for only $349.00, a 4K fast 6502 system complete with the Microsoft 8K BASIC in ROM, 32 characters/line video, a full 53 key keyboard, 32 x 32 character graphics using (note this) 256 graphics characters, runs in either BASIC or machine code, has Kansas City cassette interface, and is housed in an attractive metal cabinet with internal power supply. Wow. The graphics are easy to program, fast, and provide an equivalent screen resolution of 256 x 256 lines. Did I mention upper and lower case? The direct access video display memory is in addition to the 4K user RAM. We don't stop here. You can expand to 8K with a 4K chip kit which just plugs into the same board. Switchable connections are provided for a 300 baud RS-232 modem port and an RS-232 interfaced RO printer. And, if that were not enough, a 610 expansion board fits into the same cabinet with 8K more. And you have room on the 610 for an additional 16K, putting you up to 32. The 610 also provides interface for up to two mini floppy drives and also opens the door to a Centronics 779 printer interface and a communications port. A 620 expansion adaptor plugs into an external OSI standard 8 slot motherboard for such things as A/D, more memory and.......on and on. This is a whale of a lot for the price. Please note that the $349.00 takes you up through the first 4K only. By the way, OSI's new monitor/TV is only $115.00. Deliveries of the C1P will be limited in December but production is expected to hit full swing starting January. To be on the safe side, we are promising February. Color? Yup. Maybe about February. Prefer a 64 character line? No problem. Take a C2-4P for $599.00.

RIVERSIDE ELECTRONICS' video board MVM-1024 is another very fast video system using 128 ASCII characters with a uniquely implemented blinking cursor, reversing and blinking screen with special cursor addressing and position reading. In full gear, this rig looks like downtown Las Vegas. Just do a STA to the position and bingo, fireworks. The computer can read the position of the cursor which is constantly stored on the MVM-1024. Writing a new cursor position overwrites the old. No address lines are used for cursor position and no address space of the microprocessor is used. All operations are controlled hy writing to 3 bi-directional ports. Home? Just store a zero at ADH and ADL and zip. The MVM-1024 can be plugged directly into a KEM motherboard which mates directly to KIM with no special wiring. The KEM also accepts up to 4 S-100 boards (2 connectors come mounted) and there is space for 4K of PROM. Just connect a parallel ASCII keyboard to KEM and some power, then start punching. The video is 16 x 64 Because of all the neat contrl, this system is great for industry, labs and education. Write for complete info.

SUPERKIM by Micro Products is now available. This enhanced version of a KIM offers the KIM monitor and interfaces with 1K of RAM plus room for on-board expansion of an additional 4K and up to 16K user PROM. SUPERKIM features power-on-reset, RS-232 or 20 ma. loop, audio cassette interface, fully buffered address and data lines, a 6522 with room for 3 more, 8 individually resetable latched interrupts, a high quality hex keyboard, address and data LED display. A proto area for your kluges and all of the power supply but the transformer are included on this 11.5 x 11.5 inch wonder. SUPERKIM will run all the KIM software too. $445.00, our stock.

PET? Sure. We have the 8K version at $795.00 and they include the manual. We also carry the CMC printer interface for PET and the CMC word processor on cassette.

SYM1 is now available from Johnson Computer. We carry the power supply also. Our Microsoft BASIC is also available on cassette for SYM.

HDE FLOPPY DISK for KIM is now available in the 5" mini as well as the 8" version you have heard about. The single mini is $695.00 and the double is $1045.00. Delivery of the mini will begin at the end of January. They use the Shugart drive and HDE interface, controller and file oriented disk operating system (FODS). Comes complete with power supply, ready to plug in. The 8" drives have been popular with our industrial KIM accounts and universities. The 8" system uses the very reliable Sykes drive and sells for $1995.00 for the single and $2750.00 for the double. For a preview, manuals for either system are available for a nominal charge. Interfacing to the KIMSI and the KEM are also now available. Just plug in, boot up and that's it. Software to tie in with Microsoft BASIC is supplied at no extra cost.

HDE 8K RAM boards plug directly into a KIM-4 or can be wired direct to the expansion connector. The DM816-8 uses the 1K x 4 4804 on a 4.5 x 6.5 inch card. Industrial grade design and strict QC plus burn-in makes the DM816-8 super tough and super reliable. Card slides mount on the ends of the KIM-4 edge connectors for ridged suppot. Card guides are available at no extra cost, on request, one set per board. Price? A new low price will be effective by the time you read this line. Check with us. We have the boards in stock. Now you can afford to go to 32K, or whatever.

MOS TECHNOLOGY 6502 in OEM quantities at factory prices is available from Johnson Computer and so also for the entire 6500 family. Spec sheets on request.

OHIO SCIENTIFIC BUSINESS SYSTEMS are available from us along with business software and high performance peripherals. The proven Challenger III series offer 3 different software switchable processors allowing operation of software written for the 6502, 6800, 8080 or Z80. Up to 56K of RAM, dual single sided or dual double sided floppy disk system and an optional 74 megabyte fixed hard disk system complete with a quality CRT terminal and high performance printer is available for as low as about $14,000.00 and up, depending upon your requirements. Software is additional and starts at $975.00 for a complete standard accounting package. OSI is now introducing a new concept in business software which provides remarkable flexibility and ease of system set-up. Contact us for more details.

HAZELTINE 1400 CRT terminals in stock.
HAZELTINE 1500 CRT terminals in stock.
QUME letter quality full char. printers in stock.
CENTRONICS 779 and 703 printers in stock.
VEN-TEL accoustical couplers in stock.

MICRO TECHNOLOGY UNLIMITED's high resolution visible memory graphics opens a new challenge to mathematically inclined programmers. The "visible memory" is an 8K dynamic RAM (MTU #K-1008, $235.00), the output of which displays each bit (not byte) as it is generated by the refresh circuit. The monitor is connected directly to a video jack on the RAM board. If the K-1008 is set at 2000-3FFF, the upper left hand corner of the display is the 8 dots of data at address 2000 and is followed to the right by the 8 dots of data at 2001, continuing on with 329 dots to the line and 200 dots vertically to fill the screen. Subroutines to connect any two points with a straight line provides fast high resolution graphics which are sheer fascination. Now that MTU has developed software to interface our Microsoft BASIC for KIM, you can program the graphics in either ASSEMBLY or BASIC and use the number crunching capability of BASIC to do the heavy stuff. In addition to personal and engineering graphics, math, physics, chem and EE profs will find it an absolute joy. The K-1008 also serves as usable memory when not dedicated to graphics. A very low power 16K RAM board is available for normal storage. All this, plus the KIM, simply plugs into an MTU pre-wired rack and motherboard (K-1005, $68.00). Just connect the power to the barrier strip, connect the recorder, terminal and switches to the application connector (same as your present setup if you already have a KIM) and GO. MTU provides a test pattern program with the K-1008 and, for $25.00 more, you can get a fist full of utility programs and cassette which provides a fascinating "Swirl" program and some of the subroutines such as drawing a straight line between ay two sets of coordinates, etc. It saves a lot of key punching. Color is in the works. Interested in music? Write us for info on MTU four part harmony music synthesis on KIM.

HDE inc.

HUDSON DIGITAL ELECTRONICS INC.

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

ANNOUNCING

TWO NEW PRODUCTS FOR THE KIM-1

DM816-CC15—MOTHER BOARD

A 15 connector mother board for the KIM-1 microcomputer, with power-on reset and remote reset, stop, run and single step. Complete with all connectors in place, tested for reliable operation.

The DM816-CC15 is specifically designed for the HDE 6½" X 4½" card and cannot be used with larger formats. Intended for the serious hobbyist and professional/industrial use.

As above	$315.00
W/power supply in 19" card rack	$495.00
W/power supply in cabinet	$675.00

(Centronics printer interface available at extra cost)

DM816-MD1—MINIFLOPPY DISK SYSTEM

A small, but powerful companion to our full size floppy disk system. Includes the controller/interface card, power supply, disk drive, all connecting cables, 'FODS' software and manual. Deliveries begin February 1, 1979. Initial quantities limited.

As above	$ 695.00
As above-Dual drive	$1050.00

ORDER DIRECT OR FROM THESE FINE DEALERS:

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

PLAINSMAN MICROSYSTEMS
Box 1712
Auburn, Ala. 36830
205-745-7735

ARESCO
P.O. Box 43
Audubon, Pa. 19407
215-631-9052