KIM Rapid Memory Load/ Dump Routine

Bruce Nazarian
1007 Wright Street #3
Ann Arbor, MI 48105

This routine works well for mass entering of stuff like long programs from a hex dump or similar, where you can tell at a glance where any errors in your entries are. A few words of additional explanation about it:

For those users who would rather have a Carriage Return activate the address entry portion and the associated functions, substitute ASCII CR ($0D) at location $010E. This will do the trick and is the same as Markus Goenner's function from his TTY load routine from K.U.N. Thanks go to him for the use of some of his programming techniques.

The directions also indicate that the program will list until it senses a key pressed at the end of a line. This is true, but the user should only use one of the DATA keys on the keypad, not ST or RS.

Finally, the routine will only indicate the stopped address after the user commands RUBOUT thru his terminal. Then the KIM monitor will print the current pointer, which will be the address where it stopped dumping.

If you want the routine to present one line of hex at a time, and wait on a key depression before looping back again and printing another line, make this change:

0147 20 6A 1F JSR KEYIN (Instead of the getkey subroutine)
014A D0 FB BNE 0147
014C EA EA NOP's to fill previous coding

```
0100                          ORG $0100
0100  D8           ENTER      CLD          Clear decimal mode
0101  A9  00                  LDA #$00     Zero out the input buffers
0103  85  F8                  STA INL      Low. . . . .
0105  85  F9                  STA INH      And High. . . .
0107  20  2F  1E              JSR CRLF     Use KIM Subroutine to send functions
010A  20  5A  1E  ADDR        JSR GETCH    Input one character.. (of starting addr)
010D  C9  20                  CMP #$20     Check for go ahead.. (Insert 0D for CR)
010F  F0  05                  BEQ DATA     If yes, load address from buff in pointer.
0111  20  AC  1F              JSR PACK     If no, load character into INL,INH
0114  F0  F4                  BEQ ADDR     ...and loop back again
0116  20  CC  1F  DATA        JSR OPEN     Move INL,INH, to POINTL,POINTH..
0119  20  2F  1E  DECIDE      JSR CRLF     (Saves bytes, doesn't it?)
011C  20  5A  1E  INPUT       JSR GETCH    Now input some Hex for the code...
011F  C9  4C                  CMP #$4C     'L' (Load memory)?
0121  F0  2E                  BEQ LOAD     Yes, branch to LOAD portion (0151)
0123  C9  51                  CMP #$51     'Q' (Dump from memory)?
0125  D0  F5                  BNE INPUT    No, ignore invalid characters;Loop..
0127  A9  0F        DUMP      LDA #$0F     Set up byte counter (16 decimal)
0129  8D  7F  01              STA COUNT    stick it in $017F
012C  20  2F  1E              JSR CRLF     New line, please..
012F  20  1E  1E              JSR PRTPNT   Output the current pointer address
0132  20  9E  1E              JSR OUTSP    ...and space it...
0135  20  9E  1E  GET         JSR OUTSP    ...again...
0138  A0  00                  LDY #$00     Set up Y-Register for Indirect addressing
013A  B1  FA                  LDA (POINTL),Y Load contents of pointed address
013C  20  3B  1E              JSR PRTBYT   ...and print as two hex digits...
013F  20  63  1F              JSR INCPT    Increment the double-byte pointer
```

```
0142  CE  7F  01        DEC COUNT       Decrement the byte counter
0145  10  EE            BPL GET         And loop back if not finished yet
0147  20  6A  1F        JSR GETKEY      After 16th byte, test for end of list
014A  C9  15            CMP #$15        ...and if no key is pressed,
014C  F0  D9            BEQ DUMP        go back and output another 16 bytes...
014E  4C  64  1C        JMP CLEAR       else jump to Clear input buffs..
0151  20  2F  1E  LOAD    JSR CRLF
0154  20  5A  1E  READ    JSR GETCH       Input one character..
0157  C9  0D            CMP #'CR'       ..and if it is a carriage return..
0159  F0  F6            BEQ LOAD        ..let it function, but ignore it..
015B  C9  1B            CMP #'ESC'      ..or if it is "Escape"...go 015F
015D  D0  06            BNE STORE       ..if not, must be valid.. Store it.
015F  20  80  01        JSR STRING      ..else send '? KIM ?' prompter...
0162  4C  64  1C        JMP CLEAR       ..and clear buffers..exit load routine
0165  20  AC  1F  STORE   JSR PACK       Pack character into INL,INH
0168  D0  EA            BNE READ        If packed value is zero, skip it..
016A  20  5A  1E        JSR GETCH       Get second byte of Hex code
016D  20  AC  1F        JSR PACK        ..and pack it also..
0170  A0  00            LDY #$00        Set up for indirect addressing
0172  A5  F8            LDA INL         Bring in packed value..
0174  91  FA            STA (POINTL),Y  .. and store it at pointed address
0176  20  63  1F        JSR INCPT       Increment the double-byte pointer
0179  18                CLC
017A  90  D8            BCC READ        Branch always..
017C  EA  EA  EA        NOP             Waste some space
017F  [XX]      COUNT   [This location used to hold the variable byte cntr]
0180                    ; Subroutine "STRING" to send KIM prompter
0180                    ORG $0180
0180  A2  0C    STRING  LDX #$0C        Set up X-reg as counter
0182  BD  90  01 STRNG2  LDA TABLE,X     Get character at TABLE + X
0185  20  A0  1E        JSR OUTCH       Ship it out...
0188  CA                DEX             Decrement the counter
0189  10  F7            BPL STRNG2      Loop is not finished
018B  60                RTS             Else return to mainline when done
018C  EA  EA  EA        NOP             NOP's to fill

190   20  3F  20  TABLE   .BYTE 'SP,?,SP,
0193  4D  49  4B          M,I,K
0196  20  3F  00          SP,?,NUL,
0199  00  0A  0D          NUL,LF,CR
019C  0D                  CR'
```

Some Instructions To Help It All Make Sense:

1. This routine is set up for an I/O device of the user's choosing, as long as it is fed thru the KIM internal TTY port.. Users with other I/O will have to modify the coding to suit their particular situation.
2. The routine is self-contained on Page One and leaves all other memory free for user programs, but be prepared, as always, to re-read the routine from cassette should the stack overwrite the routine.
3. Execute as follows:
   After loading the coding, a "GO" executed at address $0100 will get the ball rolling.. your terminal should immediately execute a CR/LF sequence and will pause... Begin by typing in the four digit address you wish to start loading, or dumping from.. If you err in typing, just correct by typing in the correct address again, just like the KIM TTY monitor.. A "SPACE" after the correct address is in place will enter that address into the pointer.. The program will again send CR/LF and pause.. now, enter "L" if you wish to use the rapid load routine, or "Q" if you wish a formatted memory dump from your indicated address.. If LOAD was chosen, you may now begin entering data in two-digit HEX and the pointer will be taken care of for you automatically.. a good way to do this is

to enter two hex digits, and then space, as the routine will ignore the packed space character and only enter the valid hex... If DUMP was chosen, the routine will now commence to dump the contents of memory consecutively from your indicated address like this:

```
0200 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0210 EA EA EA ..... etc.
```

IT WILL LIST CONTINUOUSLY UNTIL YOU PRESS A KEY ON THE KIM KEYPAD AND HOLD IT DOWN AT THE END OF A LINE.. It will then stop and indicate the stopped address.

KIM-1 Tidbits

Harvey B. Herman
Chemistry Department
University of North Carolina at Greensboro
Greensboro, N.C. 27412

I have been using KIM for a number of years and wish to share programs which I have developed or modified with the readers of Compute II.

The first item is a modification to the KIM tape verify program from Issue #13 of 6502 User's Notes. This program has a small bug which affects TTY use. The TTY delay characters (CNTL30/CNTH30) are stored in $17F2 and $17F3 and are overwritten by a section (VEB) of the original verify program. Instead of the comforting KIM message on completion of the program, all I got was a meaningless chugging. The following program (origin $300) circumvents the problem by shortening the VEB section so the delay characters remain intact. I now include this in KIM Microsoft BASIC, as the User program, so I can check tapes after a SAVE.

```
                        0100 ;
                        0110 ;KIM TAPE VERIFY PROGRAM
                        0120 ;
                        0130 ;HARVEY B. HERMAN
                        0140 ;
                        0150          .BA $300
                        0160          .OS
                        0170 CHKL     .DE $17E7
                        0180 CHKH     .DE $17E8
                        0190 VEB      .DE $17EC
                        0200 LOAD12   .DE $190F
                        0210 LOADT9   .DE $1929
0300- D8                0220 VERIFY   CLD
0301- A9 00             0230          LDA #$00
0303- 8D E7 17          0240          STA CHKL
0306- 8D E8 17          0250          STA CHKH
0309- A2 06             0260          LDX #$06
030B- BD 16 03          0270 LOADP    LDA PROG-1,X
030E- 9D EB 17          0280          STA VEB-1,X
0311- CA                0290          DEX
0312- D0 F7             0300          BNE LOADP
0314- 4C 8C 18          0310          JMP $188C
0317- CD 00 00          0320 PROG     .BY $CD $00 $00
031A- 4C 1D 03          0330          .BY $4C $1D $03
031D- D0 03             0340 PATCH    BNE FAILED
031F- 4C 0F 19          0350          JMP LOAD12
0322- 4C 29 19          0360 FAILED   JMP LOADT9
                        0370          .EN
```

Item 2 is a modification to KIM Microsoft BASIC (serial number 9011) which allows one to append programs on tape to the current one (if any) in memory. Line numbers must be higher in the appended program and cannot overlap. Otherwise the only noticeable change is that one must remember to NEW before LOAD when appending is not desired. I have found this very helpful in conjunction with a renumbering program, written in BASIC (see 6502 User's Notes no. 13, p. 12).

```
                        0100 ;
                        0110 ;APPEND MODIFICATIONS TO
                        0120 ;KIM MICROSOFT BASIC
                        0130 ; SERIAL NUMBER 9011
                        0140 ;
                        0150 ;HARVEY B. HERMAN
                        0160 ;
                        0170          .BA $2785
                        0180 ;ADJUST TAPE LOAD POINTERS
2785- 38                0190 NEWLOAD  SEC
2786- A5 7A             0200          LDA *$7A
2788- E9 03             0210          SBC #$03
278A- 8D F5 17          0220          STA $17F5
278D- A5 7B             0230          LDA *$7B
                        0240 ;NAIVE HARVEY
278F- B0 02             0250          BCS SKIP
2791- E9 00             0260          SBC #$00
2793- 8D F6 17          0270 SKIP     STA $17F6
                        0280 ;ORIGINAL CODE CONTINUES
                        0290          .BA $2744
                        0300 ;ASSIGN ID 01 TO TAPES
2744- A9 01             0310          LDA #$01
                        0320          .BA $2026
                        0330 ;POINTER TO NEWLOAD
2026- 84 27             0340          .SI NEWLOAD-1
                        0350          .EN
```


I hope these programs will be found useful and plan to share other tidbits with Compute II readers in the future.