KIM-1 Tidbits

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

If you are a regular reader of Compute magazine, you may have noticed that I am an owner of several small computers - KIM, SYM, and PET. (If that sounds as if I am some sort of addict, I confess that I am.) I don't have any favorite but PET has one feature, a screen editor, that I particularly like. Mike Louder (see, Best of PET Gazette) has taken the screen editor one step further and given us a "dynamic keyboard". With this it is possible to add, modify or delete BASIC statements while a program is executing. For example, it is possible to convert a machine language program, already in memory, to a series of DATA statements in a BASIC program. These DATA statements can be used to POKE the machine language program back into memory at any convenient later time. There are several advantages to doing it this way rather than by a conventional LOAD. The BASIC program could also include instructions on how to use the machine language program. It is not difficult to write the BASIC program in such a way so as to make the machine language program relocatable. Protection of the program is easily done by POKEs from BASIC. The machine language call can be done by SYS or USR. In my opinion, converting machine language programs to DATA statements is an altogether useful application of PET's "dynamic keyboard".

The dynamic keyboard idea, as far as I know, has not been extended to KIM. Unfortunately, KIM BASIC, unlike PET, does not use a keystroke buffer which is essential to the published method. I have been brooding over this difference for some time and have finally come up with a KIM procedure (program "DATA") which is described in this article. The example shown is for converting machine language programs to data statements but could easily be adapted to other uses of the dynamic keyboard.

KIM BASIC normally gets its character input using the monitor routine, GETCH, at location $1E5A. My idea is to temporarily modify the jump instruction to the KIM subroutine (locations $2456 -$2458 in my version of BASIC). BASIC would then jump to another routine which gets its input from a buffer in high memory instead of the terminal keyboard. In this example, the buffer contains BASIC DATA statements in ASCII format. The buffer is set up in a separate step using string concatenations, ASCII conversions, and POKEs to high memory. Since the "DATA" program is slightly complicated, even confusing me if I have not seen it for a while, I have described its operation in an accompanying table. Between the comments in the program and the table, I hope readers will understand the program more easily and be able to modify it to suit their own needs.

The routine which BASIC temporarily uses for input is POKEd into memory early in the program "DATA". I have included for reference the source code ("INPUT MOD") shown in the figure. This program was assembled with Eastern House Software's Assembler/Text Editor. It makes use of an address pointer which tells what location in high memory to get the next character from. Normal BASIC input is restored when the end-of-data character ($1A) is read. However, the last character returned is not $1A, but $0F (Control O) which toggles back the BASIC output suppress switch so BASIC will print again normally. Because the high ASCII buffer contained line feed characters it was necessary to toggle the switch initially, by reading $0F as the first character, to prevent unwanted double spacing.

In summary, one starts with a machine language program and the BASIC program called "DATA". After running "DATA" one is left with a number of DATA statements and a FOR/NEXT POKEing routine which can restore the machine language program at any subsequent BASIC session. The program left after running "DATA" can be augmented with instructions and protection POKEs if desired. I am aware of two obvious restrictions. One, the machine language program cannot overlap with BASIC or this BASIC program. Two, if BASIC is in ROM another method must be used. Please let me know if other KIM owners find this program useful or if there are any questions (SASE for reply).

Steps in "DATA" Program
(Line Numbers in Parenthesis)

1. Protect high memory (63047).
2. POKE machine language program (63048, 63500 SUB).
3. Input starting and ending locations of machine language program (63050-63055).
4. POKE control O ($0F) into first location of buffer (63065).
5. Construct one data statement from each 8 bytes (63080-63150).
6. POKE ASCII characters to high memory (63400 SUB).
7. Construct POKEing program and POKE ASCII to high memory (63160-63210, 63400 SUB).
8. POKE last character ($1A) to high memory (63212).
9. Change BASIC input character subroutine (63220).
10. Unprotect high memory and erase "DATA" program (63240).
11. Input is now from high memory (with echo to terminal) until last character ($1A) is read. At this point LIST should show a series of DATA statements and a FOR/NEXT POKEing routine. This program can be SAVEd for later use.

```
INPUT MOD
                0100 ;
                0110 ;TEMPORARY MODIFICATION TO KIM BASIC INPUT ROUTINE
                0120 ;INPUT A CHARACTER FROM HIGH MEMORY
                0130 ;INSTEAD OF KEYBOARD
                0140 ;RESTORE NORMAL OPERATION WHEN $1A
                0150 ;CHARACTER IS READ FROM HIGH MEMORY
                0160 ;
                0170 ;HARVEY B. HERMAN
                0180 ;
                0190            .BA $200
0200- 18        0200            CLC
                0210 ;BUMP STORAGE POINTER
                0220 ;NOTE: PROGRAM MODIFIES ITSELF
                0230 ;      SORRY EPROM FREAKS
0201- AD 12 02  0240            LDA COUNT+1
0204- 69 01     0250            ADC #01
0206- 8D 12 02  0260            STA COUNT+1
0209- AD 13 02  0270            LDA COUNT+2
020C- 69 00     0280            ADC #00
020E- 8D 13 02  0290            STA COUNT+2
0211- AD FF 5F  0300 COUNT      LDA HIGH-MEM
                0310 ;CHECK FOR DONE($1A)
0214- C9 1A     0320            CMP #$1A
0216- F0 06     0330            BEQ END
                0340 ;KIM OUT
0218- 48        0345            PHA
0219- 20 A0 1E  0350            JSR $1EA0
021C- 68        0355            PLA
021D- 60        0360            RTS
                0370 ;RESTORE KEYBOARD INPUT
021E- A9 5A     0380 END        LDA #$5A
0220- 8D 57 24  0390            STA $2457
0223- A9 1E     0400            LDA #$1E
0225- 8D 58 24  0410            STA $2458
0228- A9 0F     0420            LDA #$0F
                0430 ;RETURN WITH CONTROL O
022A- 60        0440            RTS
                0450 HIGH-MEM   .DE $5FFF     ;-1
                0460            .EN
```



```
DATA
63000 REM KIM PROGRAM WHICH MAKES DATA STATEMEMENTS
63010 REM FROM A MACHINE LANGUAGE PROGRAM
63020 REM
63030 REM HARVEY B. HERMAN
63040 REM
63045 REM PROTECT HIGH MEMORY
63047 POKE 132,0:POKE 133,96:CLEAR
63048 GOSUB 63500:REM POKE MACHINE LANGUAGE PROGRAM
63050 INPUT "STARTING LOCATION";S
63055 INPUT "ENDING LOCATION";E
63060 N=10
63064 REM H IS HIGH MEMORY LOCATION FOR SAVE
63065 H=24576:POKEH,15:H=H+1
63080 FOR I=S TO E STEP 8
63085 A$=STR$(N)+" DATA "
63090 FOR J=0 TO 6
63100 A=PEEK(I+J)
63110 IF (I+J)=E THEN 63130
63115 A$=A$+STR$(A)+","
63120 NEXT J
63125 A=PEEK(I+J)
63130 A$=A$+STR$(A)+CHR$(13)+CHR$(10)
63140 GOSUB 63400
63145 N=N+10
63150 NEXT I
63160 A$=STR$(N)+" FOR I="+STR$(S)+" TO"+STR$(E)+
      CHR$(13)+CHR$(10)
63170 GOSUB 63400
63190 N=N+10
63200 A$=STR$(N)+" READ A:POKE I,A:NEXT I"+CHR$(13)+
      CHR$(10)
63210 GOSUB 63400
63212 POKE H,26:REM LAST CHARACTER
63215 REM CHANGE INPUT ROUTINE
63220 POKE 9303,0:POKE 9304,2
63230 REM UNPROTECT HIGH MEMORY
63240 POKE 132,0:POKE 133,144:NEW
63390 REM PICK APART STRING AND SAVE IN HIGH MEMORY
63400 FOR K=H TO H+LEN(A$)-1
63410 POKE K,ASC(MID$(A$,K-H+1,1))
63420 NEXT K
63430 H=H+LEN(A$)
63440 PRINT A$;:POKE 22,0:REM ZERO PRINT POSITION
63450 RETURN
63500 DATA 24,173,18,2,105,1,141,18
63510 DATA 2,173,19,2,105,0,141,19
63520 DATA 2,173,255,95,201,26,240,6
63530 DATA 72,32,160,30,104,96,169,90
63540 DATA 141,87,36,169,30,141,88,36
63560 DATA 169,15,96
63570 FOR I= 512 TO 554
63580 READ A:POKE I,A:NEXT I
63590 RETURN
OK
```