# A Kim-I Music File In Microsoft Basic Part 1.

Anthony T. Scarpelli
N. Windham ME

## Getting The System Together

If you have a KIM-I, don't have a printer, but do have a memory mapped video display, here's how I solved the problem of getting a software routine to cause an ASCII keyboard to act like a serial teleprinter with all the KIM-I's teletype operations. There's nothing that seems complicated about what I did, but it sure took some mental gyrations to get it working. Yet I did learn a lot about the KIM monitor routines which I'll tell you about. Also how to implement BASIC, and how to implement a Music File which I wrote for my wife. Here's the story.

I had a KIM-I up and running and was learning a lot about assembly language programming, when the opportunity of getting a high resolution video monitor for cheap came along. I bought a SWTP keyboard, and while I was at one of the computer fairs last year I purchased Microtechnology's 8K visible memory and a main frame. The price was good and it was completely compatible with the KIM. It's a dynamic memory system, but is completely invisible to normal computer use, and it has a standard video output. It works beautifully, and is fairly high in resolution with 64,000 bits as dots on the screen. Writing a "1" in a memory location lights up a dot, and, of course, a "0" turns the dot off. Microtechnology's SWIRL software routine shows the system off and provides hours of viewing enjoyment; and when company comes over it's great for showing off your computer.

Microtechnology also has a text display routine whereby, after an ASCII number is put into the accumulator, a subroutine call to the text display puts the ASCII character on the monitor screen. It provides a 53 character by 18 line display, with both upper and lower case letters. Having a software character generator gives you complete control over the configuration of the letters. For instance, I changed all my lower case letters, which I didn't need, into a table of 26 lines, dots, and other shapes for drawing on the screen. Also, the whole screen can be saved on tape. My wife was very pleased as a valentine message formed from a randomly patterned screen. Hypertape loaded the screen in under three minutes.

I also purchased from Microtechnology their bare board 16K memory, and purchased the I.C.'s and components at other sources. You can save about a hundred dollars this way, but you do have to get a few extra memory chips in case a bad one comes up and you do have to do all of your own soldering, and testing. If you go this route you might have a fault in the bare board. In the one I bought, a part of the PCB pattern wasn't etched away so I had no -5v supply. After I fixed the problem the board worked perfectly the first time running and onwards, and I have nothing but praises for the design.

Then came the job of getting my keyboard with parallel output ASCII to go serial. It turned out to be not too difficult when I found an interface in a series of articles by John Blankenship in Kilobaud. In the March '78 issue he shows how to build a parallel to serial interface for the KIM-I. It merely takes the parallel output of the keyboard, using three I.C.'s and a transistor, and the KIm's power and clock, and converts it to a serial output which is presented to the printer input of the KIM. It worked very well.

Then what? Well, here comes the hard part. In order to get the KIM to accept a printer input, you connected pins "21" and "V" on the applications connector, hit the RS button, press the RUBOUT key on your keyboard and type away. The only problem is that any ASCII characters that come in don't go anywhere except to the printer output of the same connector. The ASCII number is put into the accumulator, but how do you call up a subroutine in some other part of memory to display it? The solution wasn't too difficult. You write a little program that jumps to KIM's own GETCH subroutine which then puts the printer ASCII character into the accumulator, then jumps to the character display subroutine, then jumps back to the GETCH etc. You start out by going to the memory location where the program starts on the KIM keyboard, short the two pins together (best to get a switch to do this), hit RS, then RUBOUT, and G on the keyboard, and away you go. You're finally writing on the CRT. Now what?

With this method that's about all you can do because you are in a program of your own creation and are using KIM's ROM routines, and you have to stay there until you hit ST (stop). What I really wanted to do was have my keyboard act just like a printer: change memory, display it, and all the other things the user manual said you can do with a printer. I asked myself, how easily can this be done? More likely, how difficult is it. There were two possibilities open to me: hardware or software. My old teacher said you never learn enough by going the easy route. I didn't know-whether hardware or software was the most difficult, but I chose software. You can judge the result; I probably would have bought a printer.

To go the software route meant rewriting some of the subroutines in the KIM's ROM. To show you what routines I had to include, let's go over what happens in the KIM when you hit RS. So get out your user manual, follow the diagrams and let's go.

First look at the listings starting at 1C22 in the User Manual and also at fig. 1.

1. When the RS (reset) button is pressed the data at locations 1FFC & 1FFD, which happens to be the address 1C22, is put into the program counter. This is the entry point for the program in ROM of the 6530-002. This address is fixed and cannot be changed. It is the KIM entry via RST.

2. The first thing that happens is the stack pointer is initialized to FF.

3. Then we go to a subroutine called INITS at 1E88. In INITS, the first thing done is to put 01 into the X register and then put it into the top of the stack at 00FF.

Next, the X-index gets 00 and is stored in PADD which is the 6530-002 A ports data direction register. This is at address 1741 and makes all the ports inputs so they can accept data from TTY or KB (keyboard).

Next X-index gets 3F and is stored at 1743 which is the 6530-002 B ports data direction register, PBDD, and it makes ports PB6 and PB7 inputs, and all the rest outputs. PB7 is connected to the audio tape interface circuits and is prepared to accept program loading from tape.

Next X-index is loaded with 07 and is stored in SBD (1742) which is the data to be sent out from the 6530-002 data ports. So PB 0, 1, & 2 now have 1's on them. PB0 is for TTY data out. PB 1, 2, 3, & 4 go to the 74145 I.C.'s inputs. With a 1 on 1 & 2 and 0 on 3 & 4, all the outputs of the 74145 are high except 03. This goes out to application connections A-V. When this pin is connected to A-21 (PA0), PA0 becomes low. This indicates TTY mode.

Next decimal mode is cleared and the interrupt disable status is set. Then a return from this subroutine.

4. Next back at 1C2A, FF is stored at 17F3 (CNTH 30) which is the TTY count, and 01 is stored in the accumulator. Then SAD (1740) is tested, specifically PA0. If it is not equal to zero, that is, if it's high, the program branches to START. PA7 is tested also. This is the input from the TTY keyboard. It tests for a rubout bit. PA7 is normally a one and the program will keep on testing this input until a zero is detected and also PA0 in case the TTY mode is not wanted any more.

If a zero is detected, the accumulator is loaded with FC and the carry flag is cleared, then 01 is

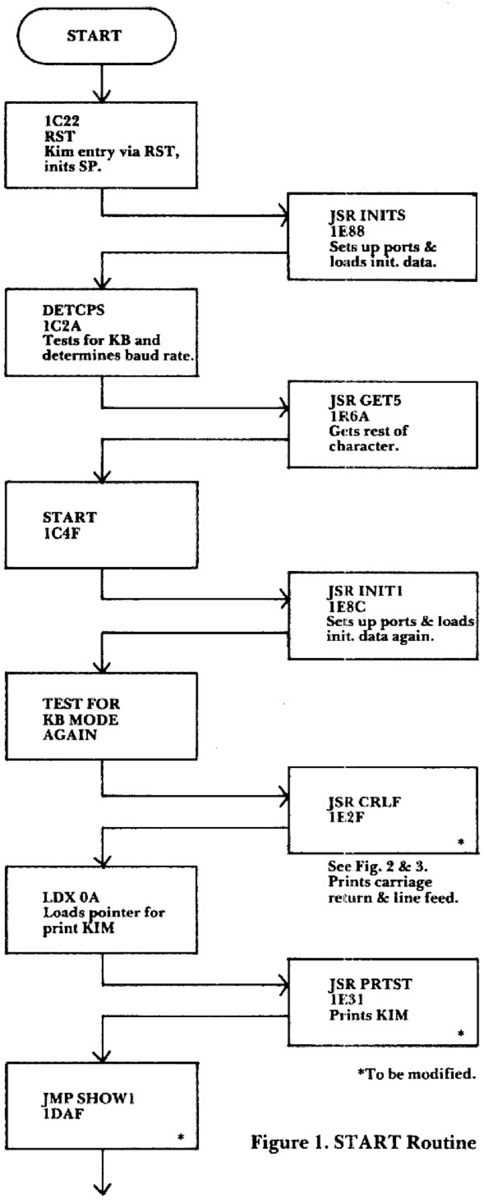

START
1C22
RST
Kim entry via RST, inits SP.
JSR INITS
1E88
Sets up ports & loads init. data.
DETCPS
1C2A
Tests for KB and determines baud rate.
JSR GET5
1E6A
Gets rest of character.
START
1C4F
JSR INIT1
1E8C
Sets up ports & loads init. data again.
TEST FOR KB MODE AGAIN
JSR CRLF
1E2F
*
See Fig. 2 & 3. Prints carriage return & line feed.
LDX 0A
Loads pointer for print KIM
JSR PRTST
1E31
Prints KIM
*
*To be modified.
JMP SHOW1
1DAF
*

**Figure 1. START Routine**

added to the accumulator (FC). If the carry flag is not set it will branch to DET 2. It will the first time around anyway. This part (DET 2) first loads Y-index with SAD (1740) and if the rubout bit is still there (a 0 at PA7) then it goes back to DET 3 and another 01 is added to the accumulator. When the accumulator reaches FF and 01 is added, the carry flag is set and CNTH30 (17F3) is incremented, it becomes 00. As long as the rubout bit is there the accumulator keeps on increasing and increases CNTH30. As soon as the bit ends the accumulator is stored in CNTL30 (17F2) and X-index gets an 08. Then the program goes to subroutine GET5 at 1E6A, where it goes to DEHALF (1EEB).

5. DEHALF first gets the high byte count time at CNTH30 and stores it in TIMH (17F4), then gets CNTL30. The accumulator and TIMH are shifted right (divides by two). If the 0 bit had a 0 the carry flag is cleared and a branch is taken to DE2, otherwise the accumulator is OR'd with 80 and it branches to DE4. If the DE2 branch was taken the carry flag has been set and next 01 is subtracted from the accumulator. The time is reduced and back with RTS. What is happening here is the keyboard baud rate in CNTL30 and CNTH30 is halved to get in the middle of the bit, then delayed one whole bit to read the next bit of the character. Cute, huh.

6. Back at 1E6D (GET2), the accumulator is loaded with SAD and the bit number 7 only is saved. 00FE is shifted right, then OR'd with the accumulator and stored in 00FE. Another delay and the process is repeated until the whole character is retreived, then another half delay, X-index is loaded with TMPX (00FD), and the accumulator gets CHAR which is the ASCII character. The accumulator is rotated left then shifted right, which gets rid of any parity bit that might be stuck on the character. Then a return to START.

7. START. First is a jump to subroutine INIT1 (1E8C) which is the same as before, it sets up the ports. The accumulator is loaded with 01, and SAD is tested again for TTY or KB mode. If there's a 1 in PA0 it branches to KB mode. If no KB mode, it then jumps to CRLF, Fig. 2 & 3, (1E2F), which

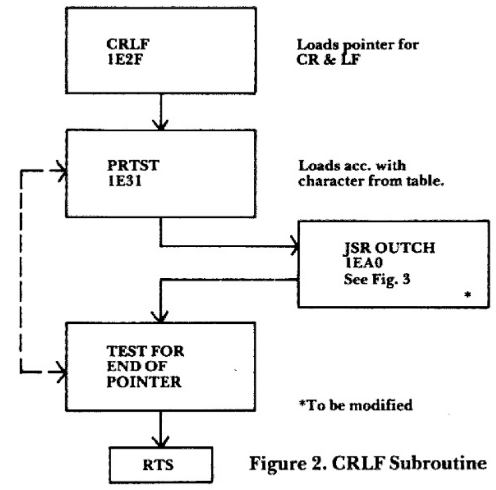


Figure 2. CRLF Subroutine

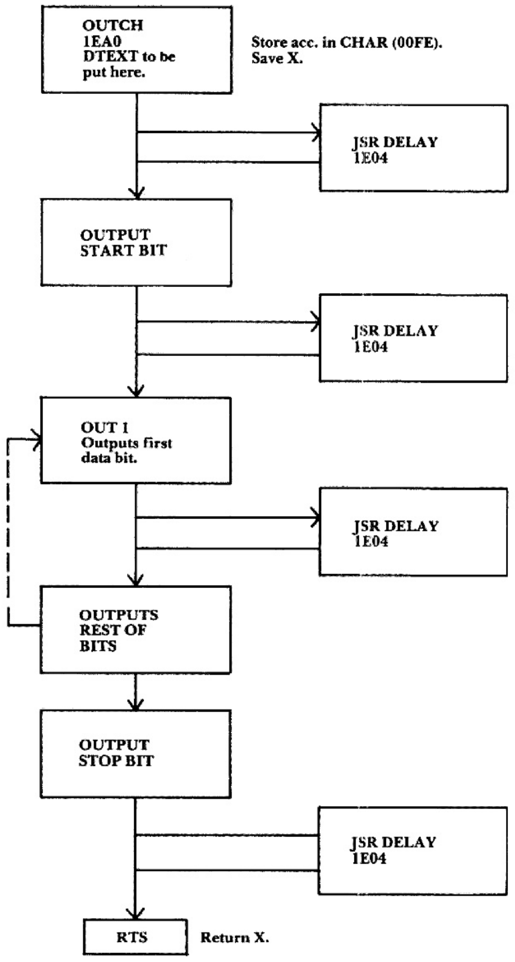


Figure 3. OUTCH Subroutine

prints a carriage return, then a line feed, then JSR PRTST prints "KIM", then jumps to SHOW1 (1DAF), Fig. 4, and then back to CLEAR, Fig. 7.

8. CLEAR. The accumulator gets loaded with 00 and is stored in INL & INH. The program tests for a character in GETCH, Fig. 8. In GETCH it stays in a loop waiting for a start bit. After the start bit, the rest of the character is retreived and loaded into the accumulator, the program then comes back, and we test for KB mode again. If no KB the character is changed into a hex number in PACK, Fig. 9, and then in SCAN, Fig. 10, the program determines if the hex number is an execute key. If not, it will get another character.

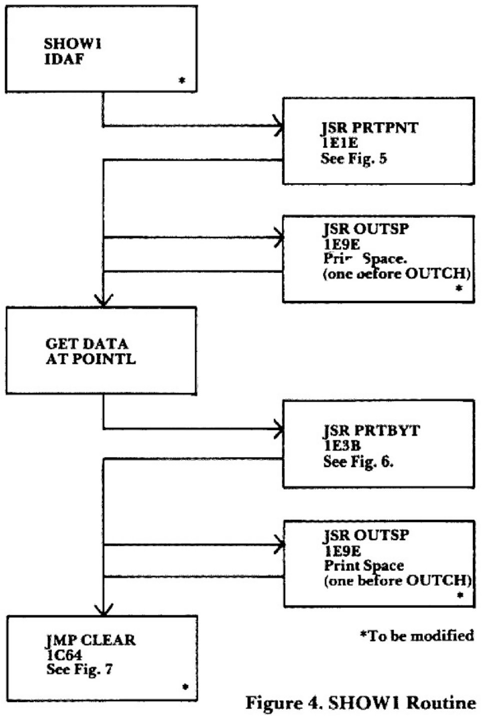
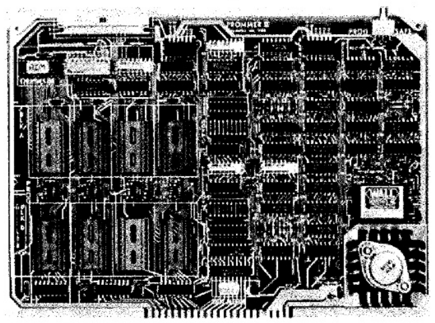


**Figure 4. SHOW1 Routine**

So this is the program I need to simulate a teletype. The problem now becomes, what are the subroutines I have to rewrite and which ones of the KIM's ROM subroutines can I use. Obviously, any part of the program that refers to a ROM address has to be rewritten, such as in a JMP. Also when the accumulator gets the ASCII character that is to be displayed, the program that does the displaying, in this case called DTEXT (the Microtechnology software routine), has to be addressed at the right point, and thus any subroutines involved here have to be rewritten. So definitely the subroutine OUTCH has to be changed to add DTEXT. We get to OUTCH from CRLF so that has to be rewritten. CRLF is addressed from START which is part of the whole RST routine. As you can see it starts to get involved. So if you go this route table I lists all the KIM ROM routines that must be rewritten. Of course in this rewriting, some branches have to be changed as well as addresses. (A SASE sent to me will get you a list of the changed addresses.)

Now my keyboard acts just as a teletype, and I can display all the teletype outputs from the KIM on the CRT. First I go to the RST program address, the one I rewrote, on the KIM display, switch to teletype mode, hit RS on the KIM, then press the rubout key on the keyboard. The SWTP keyboard doesn't have an actual rubout key, but there are two spare keys, one of which can be wired as rubout. Then I press the G key which puts me into

## SEAWELL PROMMER II -- There's Nothing Like It!



- Two independent blocks of 4 EPROM sockets
- KIM, SYM or AIM programming firmware
- Programs 1, 2 or 4K 5-Volt EPROMS: TMS2508, 2516, 2532 and 2758, 2716, 2732
- Read-Only/Deselected/Read-Program for each socket
- Program-protect toggle switch for whole board
- Provision for remoting 4 sockets
- On-board generation of programming voltage.

The Seawell PROMMER II is a general purpose EPROM tool designed for use in a development/production environment. Connects to a KIM, SYM or AIM with a Seawell LITTLE BUFFERED MOTHER motherboard, or to a SEA-1 single-board computer. The PROMMER II is all you need to read, program and execute 1, 2 or 4K 5-Volt EPROMs.

The PROMMER II allows you to put as little as 1K or as much as 32K of EPROM on the bus. You never have to give up address space to empty sockets. Each socket can be enabled to READ only, READ and PROGRAM, or can be DESELECTED entirely. Simply moving a shunt sets one block of 4 sockets to 1, 2, or 4K. The other block can be set independently.

Addresses are selected by piano-type switches on the top edge of the board. The whole board can be program-protected by a toggle switch on the top right corner of the board. A separate one-page ROM containing relocatable firmware for KIM, SYM or AIM is provided which can be set to any page in memory in either of two banks or deselected entirely. A satellite board with four sockets and program-protect switch will be available soon.

PROMMER II EPROM PROGRAMMER — $299

OTHER SEAWELL PRODUCTS:

| | | |
|---|---|---|
| SEA-1 | SINGLE BOARD COMPUTER | $495 |
| SEA-16 | 16K RAM BOARD | $250 |
| SEA-FDC8 | DOUBLE DENSITY 2-SIDED DISK CONTROLLER | $325 |
| SEA-DEBUG | HARDWARE BREAKPOINTS AND TRIGGERS | $310 |
| SEA-PROMMER II | EPROM PROGRAMMER | $299 |
| SEA-PROTO | COMPLETELY DECODED PROTOTYPING BOARD | $ 99 |
| SEA-CMOS | 8K CMOS RAM, 16K EPROM, DAY/DATE CLOCK | $475 |
| SEA-PIOB | 4 FULLY-BUFFERED 6522S | $260 |
| SEA-ISDC | 6 SERIAL PORTS WITH LOCAL PROCESSOR & DUAL PORT RAM | $595 |
| SEA-LBM | LITTLE BUFFERED MOTHER FOR KIM, SYM, AIM, SEA-1 | $199 |
| SEA-MOTHERS | 4-SLOT MOTHERBOARD | $50.00 |
| | 10-SLOT MOTHERBOARD | $125 |

ALL PRODUCTS ASSEMBLED. NO KITS

ADDITIONAL CONFIGURATIONS
MASTERCHARGE/VISA ACCEPTED

SEAWELL MICROSYSTEMS

P.O. Box 30505,
Seattle, Washington 98103, U.S.A.,
(206) 782-9480

the RST program, (rewritten). When the rubout key is pressed again the CRT will display "KIM" and also the address of the RST program; now we are as a teletype with all its functions. Simple, wasn't it.

Next time I'll go into the actual file program that creates a music file, and then can search it for any of a number of subjects.

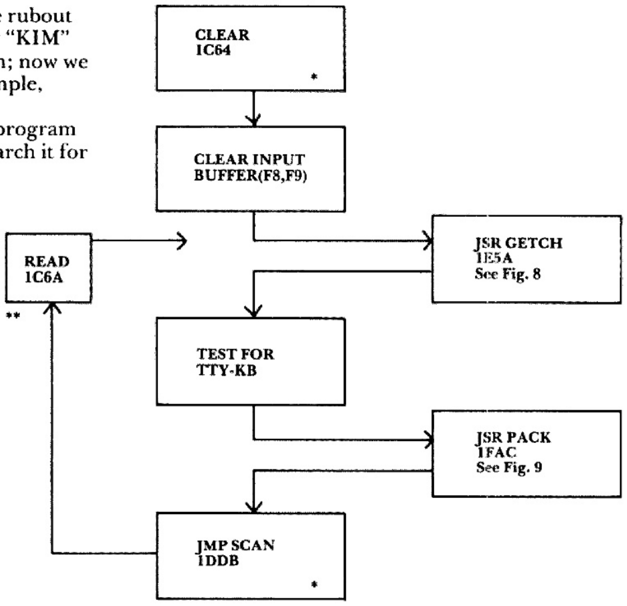


Figure 7. CLEAR Routine

*To be modified
**Re-entrance from SCAN

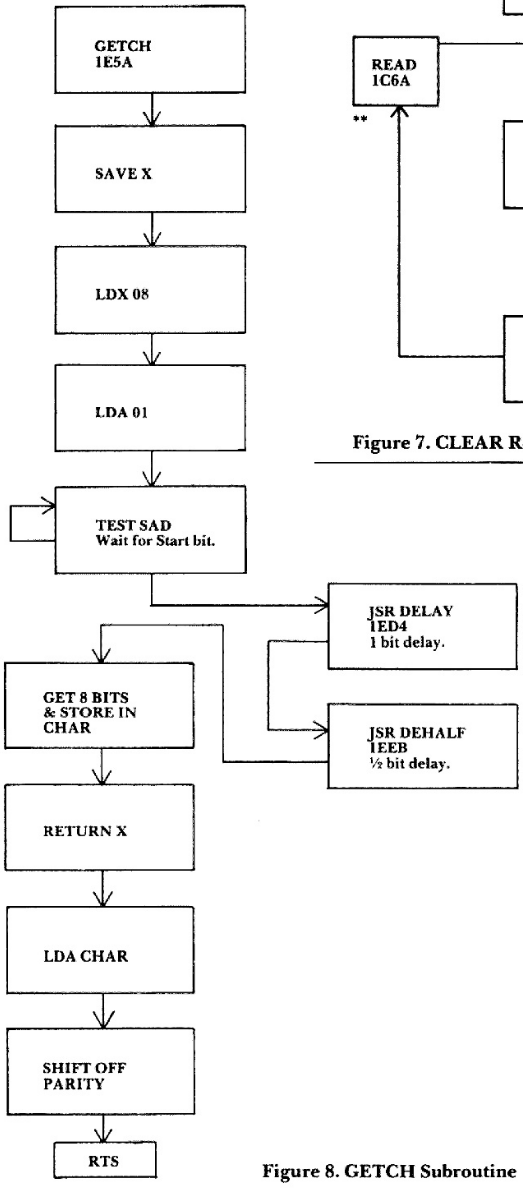


Figure 8. GETCH Subroutine

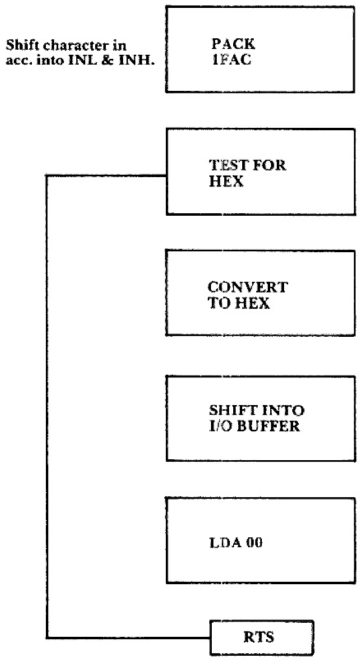


Figure 9. PACK Subroutine

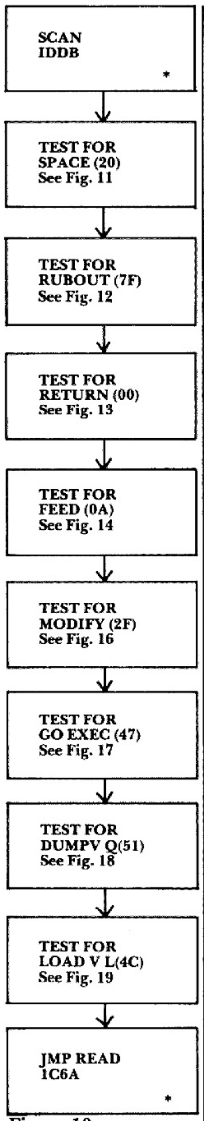
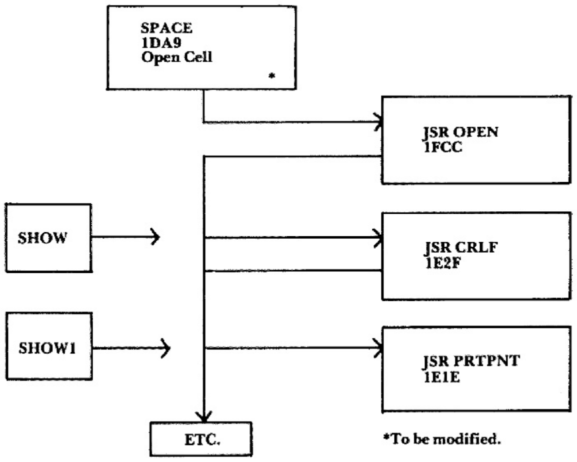

SCAN
IDDB
*
TEST FOR SPACE (20) See Fig. 11
TEST FOR RUBOUT (7F) See Fig. 12
TEST FOR RETURN (00) See Fig. 13
TEST FOR FEED (0A) See Fig. 14
TEST FOR MODIFY (2F) See Fig. 16
TEST FOR GO EXEC (47) See Fig. 17
TEST FOR DUMPV Q(51) See Fig. 18
TEST FOR LOAD V L(4C) See Fig. 19
JMP READ 1C6A
*

Figure 10. SCAN Routine



*To be modified.

Figure 11. SPACE Routine

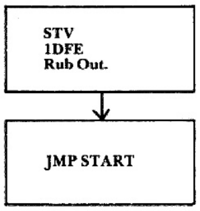


Figure 12. STV Routine

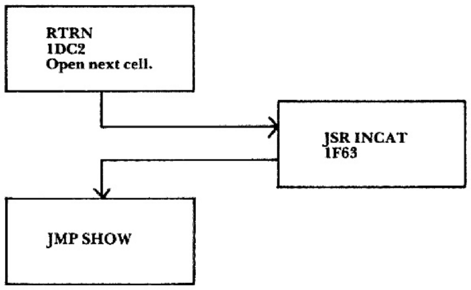


Figure 13. RTRN Routine

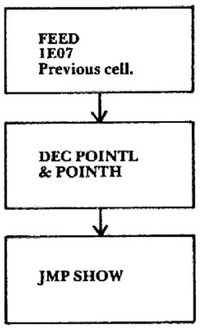


Figure 14. FEED Routine

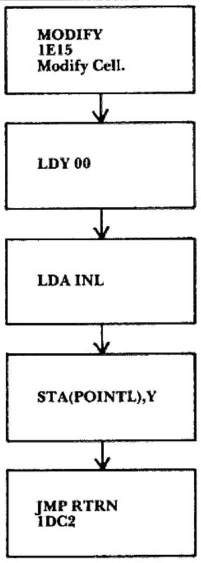


Figure 16. MODIFY Routine

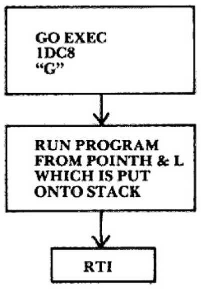
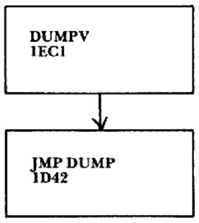


Figure 17. GOEXEC Routine



Dump from open cell to hi limit. Prints out memory.

Figure 18. Dump "V" Routine

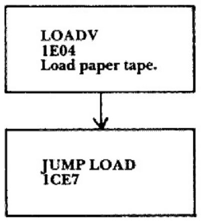


Figure 19. Load "V" Routine

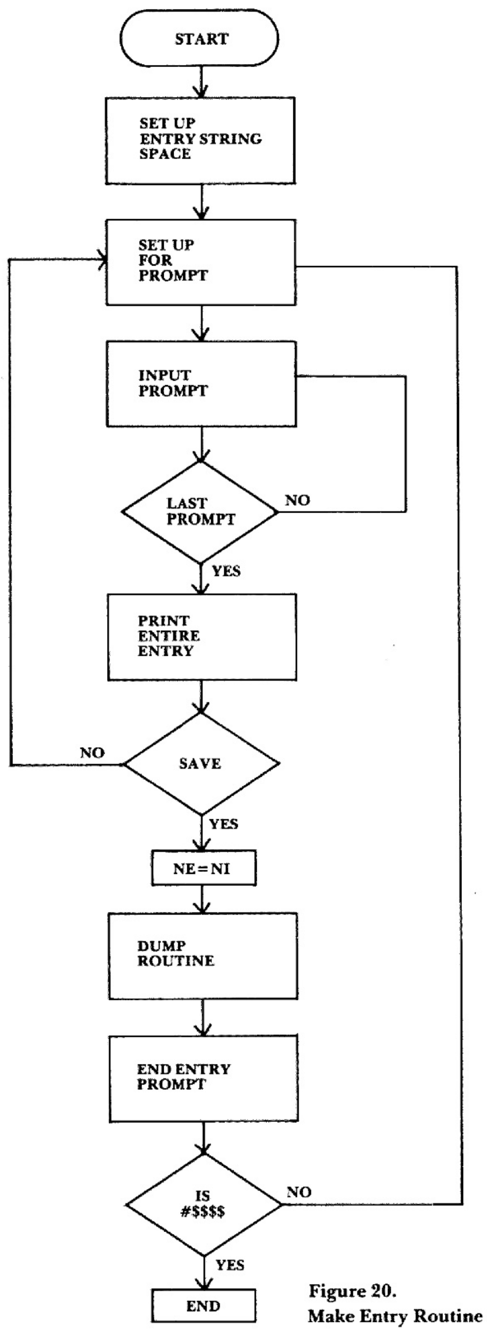


**Figure 20.**
**Make Entry Routine**

**Assembly Language Program for Cassette DUMP & LOAD**

```
0300 A5 01      DELAY  LDA DELTIM Load delay
  02 85 EE             STA TIMEA    value.
  04 A9 30      DECB   LDA 30     Load
  06 8D 04 17          STA 1704     timer.
  09 2C 07 17   TEST   BIT 1707   Test timer.
  0C 10 FB             BPL TEST   Branch if not run out.
  0E C6 ED             DEC TIMEB  Reduce time value.
  10 D0 F2             BNE DELB   Start again.
  12 C6 EE             DEC TIMEA  Reduce delay value.
  14 D0 EE             BNE DELA   Branch if not done.
  16 60                RTS        Return.
0317 A9 02      TWRITE LDA #02    Turn tape on.
  19 10 02             BPL TAPE
  1B A9 01      TREAD  LDA #01    Turn tape off.
  1D 4D 03 17   TAPE   EOR 1703
  20 8D 03 17          STA 1703
  23 60                RTS        Return.
0324 20 17 03   WRITE  JSR TWRITE Turn tape on.
  27 20 00 03          JSR DELAY  Delay for tape speed.
  2A 20 00 02          JSR HYPER  Record in hypertape.
  2D 20 17 03          JSR TWRITE Turn tape off.
  30 20 8C 1E          JSR INITI  Open ports again.
  33 60                RTS        Return.
03D5 20 1B 03   READ   JSR TREAD  Turn on tape.
  D8 20 36 03          JSR LOADT  Load tape.
  DB 20 1B 03          JSR TREAD  Turn off tape.
  DE 20 8C 1E          JSR INITI  Open ports again.
  E1 60                RTS        Return.
```

Note: HYPER is taken from The First Book of KIM page 119 relocated to address 0200.

LOADT is taken from the KIM-1 User Manual Program listing page 6 relocated from address 1871-1931 to 0334-03D4.

If you wish to use the same routines in the same addresses as I did, send a SASE and I'll let you know what locations have to be changed in those listings to get it to run right. ©

DISK DRIVE WOES?
PRINTER INTERACTION?
MEMORY LOSS?
ERRATIC OPERATION?

**Don't Blame The Software!**

Power Line Spikes, Surges & Hash could be the culprit!
Floppies, printers, memory & processor often interact! Our unique ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

- **ISOLATOR (ISO-1) 3 filter isolated 3-prong sockets; integral Surge/Spike Suppression; 1875 W Maximum load, 1 KW load any socket ........ $62.95**
- **ISOLATOR (ISO-2) 2 filter isolated 3-prong socket banks; (6 sockets total); integral Spike/Surge Suppression; 1875 W Max load, 1 KW either bank ........ $62.95**
- **SUPER ISOLATOR (ISO-3), similar to ISO-1 except double filtering & Suppression ........ $94.95**
- **ISOLATOR (ISO-4), similar to ISO-1 except unit has 6 individually filtered sockets ........ $106.95**
- **ISOLATOR (ISO-5), similar to ISO-2 except unit has 3 socket banks, 9 sockets total ........ $87.95**
- **CIRCUIT BREAKER, any model (add-CB) ........ Add $ 8.00**
- **CKT BRKR/SWITCH/PILOT (-CBS) ........ Add $16.00**

Master-Charge, Visa, American Express
**Order Toll Free 1-800-225-4876**
(except AK, HI, MA, PR & Canada)

*ESP Electronic Specialists, Inc.*
*171 South Main Street, Natick, Mass. 01760*
Technical & Non-800: 1-617-655-1532