SEPT & NOV 1977 KIM-1/6502 USER NOTES ISSUES 7 & 8

We're beginning to feel like nomads here at the USER NOTES! As you can see from the new return address we've moved again. I'd like to thank you for your patience. I've decided to make this a double issue to help make up for the delay. Hope you notice our new mailing labels. KIM is now doing a little work for the newsletter (it's only fitting, right?). See the "SOFTWARE REVIEW" for mor info on this godsend of a software package.

## ATTENTION NEW SUBSCRIBERS!!!!!!!!

Unfortunately, we are completely sold out of back issues to the newsletter. If you signed up for issues 1 thru 6 you are automatically being set up for issues 7 thru 12 instead. Plans for reprinting have not been finalized. As soon as things are nailed down as far as price and availability are concerned, that info will be passed along in the NOTES.

## 57109 CALCULATOR CHIP AVAILABILITY

In the last issue of USER NOTES, the new RPN calc. chip from NATIONAL was mentioned as a idea for a KIM interface. It is advertised as being available from TRI-TEK INC., 6522 N 43rd Ave., Glendale, Az 85301.

The price quoted is $21.92 for the chip and data sheets or $2.00 for the data sheets alone.

## FROM THE FACTORY

## AVAILABILITY OF MEMORY & MOTHERBOARDS

As you know, the KIM-2 and 3 (4K and 8K RAM cards) have been discontinued. The KIM-4 Motherboard is back on the production list and should be available in December. The KIM-3A, long awaited 8K replacement board, will be delayed indefinately.

However, don't despair!!! It is possible to adapt boards of the S-100 genre to the KIM-4 motherboard. In fact, an application note describing one such adaptation is available from MOS TECHNOLOGY. This app. note describes the mechanical and electrical interface necessary to add a KENT-MOORE ALPHA-VIDEO or their 4K RAM board to the motherboard. These two particular S-100 boards are fully assembled and tested and worked well.

Other S-100 boards could also be adapted, but due to the wide variance of signal requirements necessary for the seemingly "standard" bus structure, all other adaptations are left up to the cleverness of the user.

KIM-1 USER NOTES are published bi-monthly by Eric C. Rehnke, 109 Centre Avenue, W. Norriton, Pa. 19401. Subscription rates are $5.00 for six issues (U.S. & Canada)..$10.00 for six issues elsewhere. No part of the USER NOTES may be copied for commercial purpose without the expressed written permission of the publisher. Articles herein may be reprinted by hobby or club newsletters as long as proper credit is given and the publisher is provided with a copy of the publication.

## SOFTWARE REVIEW

by the editor

.............Get "HELP" from the COMPUTERIST......

HELP is a series of application programs which include a mailing list handler, a text editor and printing package, and an information retrieval program, which run on the naked KIM. I used the mailing list package. All I added was another cassette, a couple of TTL-controlled relays, and, of course, a hard-copy terminal (which is needed for all three packages). But, come to think of it, you could probably get away with using one of the low cost impact printers out on the market.

Anyway, the software is really excellent. "HELP" is actually an interpreter-style parameter-passing language which is very well documented and worth every penny of the $15.00 price just to see how it works! It would seem fairly straightforward to adapt this style of mini-interpreter to about any kind of application, such as; data collection, text editing, word processing, game playing, disc-file management, etc.

All sorts of neat things can be done with a little imagination!!

"HELP" REALLY IS IMPRESSIVE!!!!!!!!Seeing KIM doing some useful work for the newsletter is a thrill that just can't be described!!!

I highly recommend that you get more info on the "HELP" mailing list package as well as the rest of the "HELP" packages. Each are $15.00.

For the latest information, write: The COMPUTERIST, PO Box 3
S. Chelmsford, Ma 08124

P.S. Ask for their complete catalog and a copy of their simplified 6502 op-code table.

••••••••••••••••••••••

## 6502 vs. Z80

Want to know which chip comes out on top? Then get a copy of KILOBAUD #10. Turn to page 20 and read the article.

Z80 Freaks---eat you hearts out !!!

••••••••••••••••••••••••

## ...GOOD GUYS REALLY COME THROUGH !!!

In issue #6, I asked for volunteers who would be willing to help out other members of the group by answering questions etc. through the mail. Here are the first of the "good guys" DON'T FORGET TO SEND A SELF-ADDRESSED-STAMPED-ENVELOPE with your correspondence so our friends don't go broke.

Bruce Davidson, Box 1738, Bismark, ND 58501

Mike Jerabek, c/o University of New Hampshire, Physics Dept., Demeritt Hall,
Durham, N.H. 03824 (SOFTWARE)

Stan Bowling, 828 N. 31St., Colorado Springs, Colo. 80904 (HARDWARE & SOFTWARE)

Alan Jorgensen, 14007 N. 35th Drive, Phoenix, Arizona 85023

John Fallisgaard, Apt. #604, 1101 S. W. Pkwy., College Station, Tx. 77840
(HARDWARE & SOFTWARE)

Thomas Bray, Apt. #5, 1945 N. Oakland Ave, Milwaukee, Wisc. 53202

If your looking for a bit of fame (not much fortune) then add your name to our growing list of "GOOD GUYS".

Eric.....

••••••••••••••••••••

Philip A. Wasson
9513 Hindry Pl.
Los Angeles, CA 90045

# TRACE

With this program and about $2.00 worth of hardware you can see displayed on an oscilloscope screen, all the registers in the 6502 and three consecutive memory location starting at the address contained in the registers.
They are displayed in the following format:

```
  PC XXXX XX XX XX
  SP 01XX XX XX XX
     XXXX XX XX XX
  NV bdIZC X  Y  A
  xxxxxxxxXX XX XX
```

The first line shows the label PC, indicating the program counter, followed by the the address contained in the PC, followed by the contents of three consecutive address, starting at the value of the PC.
The second line shows the stack pointer in the same format.
The third line shows a user definable address and displays it in the same format as above.
The fourth line shows labels for the bits of the P register and for the X, Y, and A registers.
The last line shows the contents of the registers.

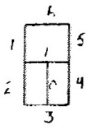
The program consists of a software driven graphics generator, a display formatter, and a monitor. It resides in $0200-$03FF.

MEMORY ALLOCATION:

| | |
|---|---|
| 03EB-03FE | SEGMENT FORMAT TABLE |
| 03E0-03EA | CHARACTER FORMAT TABLE |
| 03B1-03DF | LINE FORMAT ROUTINE |
| 03A9-03B0 | PATCH AREA |
| 0360-03A8 | DISPLAY ROUTINES |
| 0303-035F | DSPREG |
| 0270-0302 | MONITOR |
| 022B-026F | HEADING TABLE |
| 021B-022A | EXIT ROUTINE |
| 020D-021A | PATCH AREA |
| 0200-020C | INITIALIZATION OF NMI VECTOR |

Here are the locations of several useful subroutines:

0303 DSPREG - Displays all registers.
0360 OUTBYT - Displays a byte in A.
036B OUTCHR - Displays a symbol if bit 7 of the accumulator is off. Symbols displayed are: 0,1,2,3,4,5,6,7,8,9,0, A,b,C,d,E,F,o,i,P,B in order of the numeric value of the five low order bits of the accumulator.
If bit 7 is on, a vector is drawn in one of fifteen direction, depending on the value of the low order bits. Bit 0 is used for beam blanking. Bits 1 and 2 along with bits 3 and 4 indicate the new relative vertical and horizontal position, respectively.
Bits 5 and 6 are vertical and horizontal reset, respectively.
0374 OTSEGS - Displays a symbol in the following 8 segment display format, with the bits in the accumulator indicating the corresponding segments to be displayed.



038B NEWLN - Returns beam to left margin and down one lin
038F NEWPG - Returns beam to top left margin.

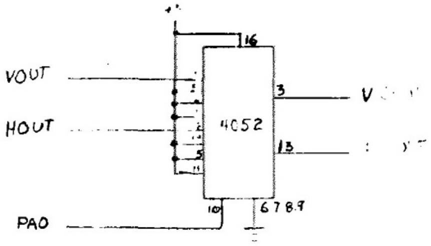
$1701 MUST BE SET TO $FF BEFORE CALLING THESE ROUTINES!

CONSTRUCTION AND USE

Construction layout of the oscilloscope driver circuitry is not critical, but leads should be kept as short as possible. It is important that the power supply be well regulated for a stable display. A 309 or 7805 type regulator is adequate.

Some users may want to use a CMOS 4555 instead of the TTL logic.

If your oscilloscope does not have a Z axis input, the following circuit is suggested. This circuit deflects the beam off the screen during the blanking period.



To use the program, connect A-15 to E-6 on the KIM connectors and begin execution at $0200. This sets the NMI vector to $0270. Now, when you press the ST key, you will be in the TRACE monitor. This monitor is just like the KIM except it is always in single step mode (even though the SST switch is off!) and when AD is pressed, it is put in address mode and the address is decremented by one. To return to the KIM, press RS.

Set $ED and $EF to the address you want to monitor. This address and it's contents will then be displayed continuously on the third line of the display.

Set your oscilloscope to x-y input mode and the horizontal and vertical attenuators to about .2V/cm DC. Connect the x, Y, and Z inputs to the driver circuit. Adjust the beam intensity for optimum character definition.

You will notice that the KIM display is dimmer than usual and there is some flicker of the displays, about 16 frames per second. Also the display on the scope may be slanted. To correct this, adjust the 50K trim pots for horizontal lines and vertical margins.

If the scope display appears to be written in hieroglyphics, the beam blanking may need to be inverted. To do this, set $039C to $01.

MODIFICATIONS

The trick to single step operation without using the SST switch is in the interupt exit routine. This routine sets the timer to give an NMI one clock cycle after the RTI is completed. This is part way into the next instruction to be executed. Since all instructions take at least 2 cycles, and the interupt is inhibited until the instruction is complete, only one instruction is executed before the NMI occurs. Thus a single step function is performed.

```
21B AD 03 17 INTEX LDA PBDD
21E 29 7F          AND =$7F
220 8D 03 17       STA PBDD
223 A9 28          LDA =$28
225 8D 0C 17       STA CLK1TI
228 4C C8 1D       JMP GOEXEC
```

more...

## TRACE (contd)

In bebuging large programs with many loops it is desirable to use conditional tracing. To do this, the user must write a routine to test the desired conditions to be traced. Locations $0287 and $0288 are set to the address of the test routine (low order byte first, of course). If the condition is met, the test routine exits with a JMP $1F88 (INITS). Otherwise, exit with:

```
          PLA
          PLA
          JMP $021B
```

EXAMPLE: Trace if X is less than 2 OR A=0.

```
TEST    LDA $F5    GET VALUE OF X
        CMP #2
        BCC TRUE   SINGLE STEP IF X IS LESS THAN 2
        LDA $F3    GET VALUE OF ACCUMULATOR
        CMP #0
        BEQ TRUE   SST IF A=0
FALSE   PLA
        PLA
        JMP $021B EXECUTE NEXT INSTRUCTION
TRUE    JMP $1E88 RETURN TO TRACE MONITOR
```

IF YOU ARE USING CONDITIONAL TRACING, IT IS NECESSARY TO ENTER THE TRACE MONITOR AT $0289, INSTEAD OF BY THE ST KEY!

EXAMPLE: Press RS, AD, 0, 2, 8, 9, GO
Now set address where tracing is to begin and press GO.

To return to normal tracing, set $0287 to $88 and $0288 to $1E.

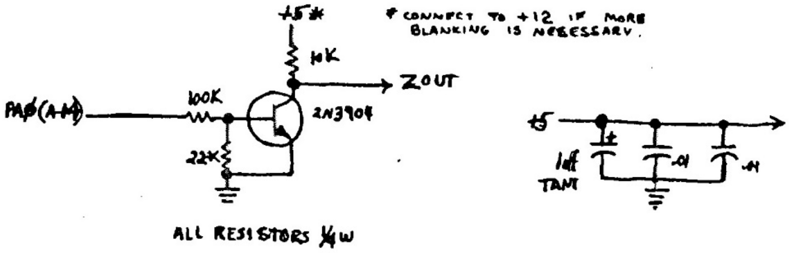
The following routine executes a program in "slow motion", about one instruction per second, and displays all the registers on the oscilloscope screen.

```
200 A2 11     SLOMO LDX #$11  ;SPEED CONSTANT
202 8E 0F 02 LP     STX SAVX+1
205 20 03 03        JSR DSPREG
208 20 6A 1F        JSR GETKEY
20B AA              TAX       ;SET FLAGS IN P REG
20C F0 0A           BEQ TOMON
20E A2 00     SAVX  LDX #*-*
210 CA              DEX
211 D0 EF           BNE LP
213 68              PLA
214 68              PLA
215 4C 1B 02        JMP $021B ;TO EXECUTE ONE INSTRUCTION
218 4C 88 1E TOMON  JMP $1F88 ;RETURN TO TRACE MONITOR
```

To start SLOMO, set $0287 to $00 and $0288 to $02 with KIM . Enter TRACE monitor by starting execution at $0289. Then set address where tracing is to begin and press GO.

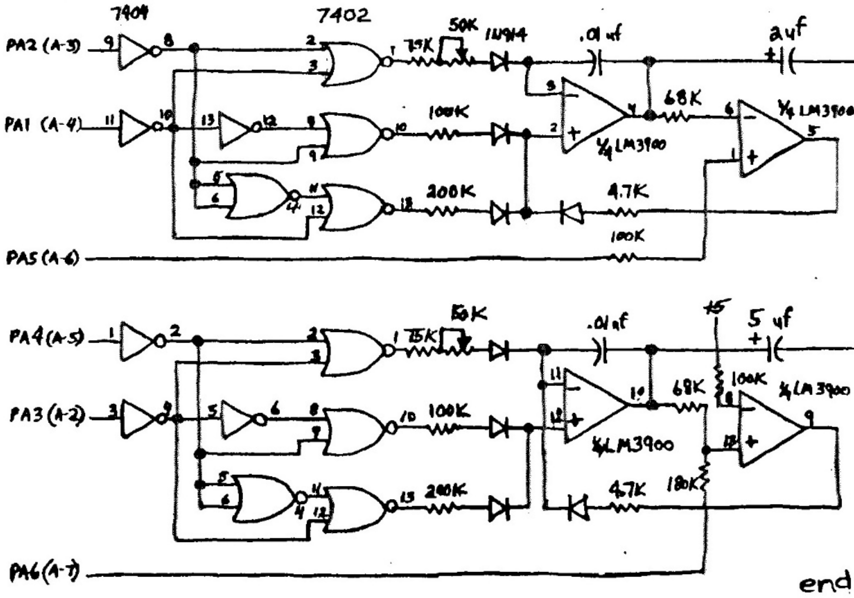
To return to TRACE monitor, press 0 key.

To resume SLOMO, press GO.



## HEX DUMP OF "TRACE"

```
ADDR    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200   A9 70 8D FA 17 A9 02 8D FB 17 4C 89 02 00 00 00
0210   00 00 00 00 00 00 00 00 00 00 00 AD 03 17 29 7F
0220   8D 03 17 A9 23 8D 0C 17 4C C8 1D 12 0C 13 05 12
0230   13 13 13 13 86 85 85 8F 8F 85 85 86 88 84 87 87
0240   8D 8D 86 88 13 0B 0D 84 91 98 88 87 87 88 93 91
0250   84 88 91 86 99 8D 8D 99 96 88 0C 13 84 8F 8F 98
0260   8D 8D 86 88 13 13 84 8F 86 65 8D 86 88 13 13 0A
0270   85 F3 68 85 F1 68 85 EF 85 FA 68 85 F0 85 FB 84
0280   F4 86 F5 BA 86 F2 20 88 1E 20 8C 1E 20 03 03 20
0290   19 1F D0 F5 20 03 03 20 19 1F F0 F8 20 19 1F F0
02A0   F3 20 6A 1F C9 15 10 E1 C9 14 F0 4C C9 10 F0 2C
02B0   C9 11 F0 34 C9 12 F0 37 C9 13 F0 39 0A 0A 0A 0A
02C0   85 FC A2 04 A4 FF D0 0A B1 FA 06 FC 2A 91 FA 4C
02D0   D7 02 0A 26 FA 26 FB CA D0 EA F0 10 A5 FA D0 02
02E0   C6 FB C6 FA A9 01 D0 02 A9 00 85 FF 4C 89 02 20
02F0   63 1F 4C 89 02 4C 1B 02 A5 EF 85 FA A5 F0 85 FB
0300   4C E4 02 20 8F 03 A9 FF 8D 01 17 A2 00 A5 EF 85
0310   F6 A5 F0 85 F7 20 B1 03 A5 F2 85 F6 A9 01 85 F7
0320   20 B1 03 A5 ED 85 F6 A5 EE 85 F7 20 B1 03 A0 3C
0330   BD 2B 02 20 6B 03 E8 88 D0 F6 20 8B 03 A5 F1 A0
0340   08 2A 48 A9 10 90 02 A9 11 20 6B 03 68 88 D0 F1
0350   A2 03 B5 F2 20 60 03 A9 13 20 6B 03 CA D0 F3 60
0360   48 4A 4A 4A 4A 20 6B 03 68 29 0F 30 2A 8E 89 03
0370   AA BD EB 03 8D FF 03 A2 08 BD DF 03 30 04 2E FF
0380   03 2A 20 97 03 CA D0 F1 A2 03 60 A9 46 D0 02 A9
0390   60 86 FD A2 10 D0 04 86 FD A2 03 49 00 8D 00 17
03A0   CA D0 FD 8E 00 17 A6 FD 60 00 00 00 00 00 00 00
03B0   00 A0 03 BD 2B 02 20 6B 03 E8 88 D0 F6 A5 F7 20
03C0   60 03 A5 F6 20 60 03 8E DE 03 A2 03 A9 13 20 6B
03D0   03 B1 F6 20 60 03 C8 CA D0 F2 20 8B 03 A2 03 60
03E0   90 02 88 9E 08 02 0C 03 03 08 02 FC 30 6E 7A B2
03F0   DA DE 70 FE FA F6 9E CC 3E CE C6 1E 01 E6 00 00
```



end

P[illegible]p A. Wasson
9[illegible] Hindry Pl.
Los Angeles, CA 90045

## TWO "NEW" INSTRUCTIONS FOR THE 6502

Have you ever wondered if those undefined op codes for the 6502 do anything? Well, there are at least two "new" instruction that I have discovered. First let me warn you that they are undocumented and are subject to change by the manufacturer. Also they are a little strange.

The first is op code 7E which I have given the nmemonic DXE which stands for "Decrement if index register X Equals zero". The only address mode is absolute. The use of the DXE only seems to effect the N flag, which appears to be undefined but depends on the value of X.

The second op code is 9E. I have given it the mnemonic SXNE, which stands for "Set effective address to one if index register X does not equal zero, otherwise set to zero". The only addressing mode is absolute indexed by Y. It does not appear to set any flags.

There also appear to be some redundant op codes, such as, 66=C6, 6A=0A, etc. My search has by no means been exaustive so there may still be some more undiscovered instructions.

The date code on my 6502 is 0676 so it doesn't have the ROR instruction. If the 6502 is microprogrammed later versions may respond differently to these op codes.

Some comments + corrections from - Mike Firth, 104 N. St. Mary, Dallas, TX 75214

Before going to the main point of my letter, I want to say that I have my programming for my Polymorphics Video Board running nicely. It has the built in ability (by changing a flag) to work with 32 or 64 character lines, allowing for the wiring scheme of the Poly board (ie. ignore address line 5 for 32 characters). The programming includes all of the screenread functions, home, line feed, carriage return, blank screen, backspace, forward curser (without changing characters) up and down curser. For my own purposes I will be working on an editor (or adapting HELP which I have bought but not yet received) to permit character editing and writing of the screen to tape and loading from tape to the screen.

I am about to buy the 8K base 2 (advertised in ON LINE) S-100 board, which is $125 for the slower speed I can use and is by far the cheapest I have seen. Will let you know.

### MORE TRIAC

It may be a bit late, but I do have to point out a couple of things about the notes on running a triac from KIM in issues 3 and 4. The original (#3,p.8) works much better if the load is attached to MT2 and the plug or power supply is to MT1 (in other words, exchange the labels at the right of the bottom diagram on page 8.)

I am somewhat surprised the circuit shown in the diagram in KUN4 (p.6) works at all, for several reasons. First, I believe the resistance connection from the photocell (shown as 10K) should go to MT2 and not beyond the load.

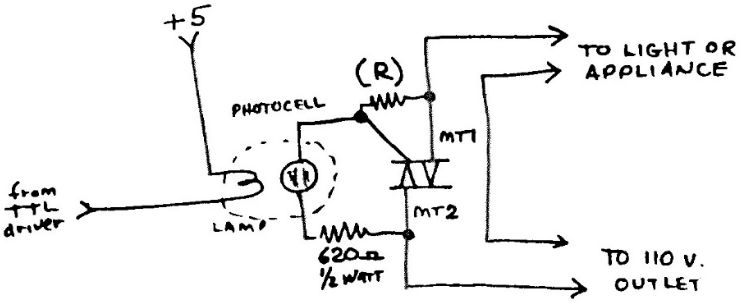
The flicker that is mentioned can come from either of two sources, both of which should make the circuit work poorly. The Radio Shack CdS cells that I purchased (and have used for other projects) have a very slow decay time, on the order of a second. Secondly, making an incandescent lamp respond in something like a single cycle (120 per second) is very unlikely. Therefore, the pulses are modulating the lamp just above and below the trigger brightness needed for the triac. Well, sometimes, due to slight shifts in the characteristics of the lamp and the cell and the triac the trigger signal will either come late in the cycle or just miss for several cycles causing flicker. (Example, lamp heats photo resister, changing resistance, lamp is pulsed less often, unit is cooler, slowly the resistance changes, besides the light effect.) I think examination of the Triac wave forms will show a very sloppy output that may harm some motors. Take care.

4

MORE ON THE TRIAC FROM: G. THOMPSON, 39 JUDSON ST. ROCHESTER, N.Y

"HERE IS A REVISION ON CASS LEWART'S TRIAC INTERFACE (#3, P. 8) THAT IMPROVES SHUT OFF.

IWAS RUNNING A 25W. BULB AND NOTICED THAT SHUT-OFF WAS NOT IMMEDIATE-THE BULB WOULD GLOW AT HALF BRILLIANCE FOR A SECOND OR SO-THEN EXTINGUISH. A SCOPE SHOWED THAT THE TRIAC WAS ACTING LIKE AN SCR DURING THIS DIMMED PERIOD, THAT IS, HALF-WAVE INSTEAD OF FULL. THE SMALL RESISTOR (R) WAS ADDED AFTER STUDYING RADIO SHACKS CIRCUITS FOR DIACS AND TRIACS. IT WORKS ONA 25W. BULB, AN AQUARIUM PUMP , AND A 1/20 HP WATER PUMP!



10 < R < 50Ω depending on load

**************************************************

Charles C. Ohsiek<br>
Box 853<br>
Patchogue, NY 11772

This code allows writing an ID on the audio cassette tape prefixing the data SUPERTAPE writes out. This ID can then be shown by VU-TAPE, or ignored by the KIM-1 tape monitor.
The ID consists of one byte, or two hex characters, at address 17F9; these two hex characters MUST BE IDENTICAL; i.e., 11, 77, AA, etc. NOT 01, 07, etc.; otherwise it cannot be viewed properly on LED's. This allows fourteen different ID's before duplicating.

Relocatable

```
(01BF C3 03 7E   END OF  SUPERTAPE)
 01C2 A0 BF      START   LDY    #$BF      Set directional
 01C4 8C 43 17           STY    PBDD      .registers
 01C7 A2 08              LDX    #$08      Send 8
 01C9 A9 16              LDA    #$16      .sync
 01CB 20 61 01           JSR    HIC       ..characters
 01CE A9 2A              LDA    #$2A      Send
 01D0 20 88 01           JSR    OUTCHT    .asterisk
 01D3 AD F9 17           LDA    ID        Setup to send
 01D6 A2 64              LDX    #$64      .100
 01D8 86 E0              STX    TIC       ..ID characters
 01DA 48         LP      PHA              ..save character
 01DB 20 70 01           JSR    OUTBT     ....send it
 01DE 68                 PLA              .....bring it back
 01DF C6 E0              DEC    TIC       Decrement counter
 01E1 D0 F7              BNE    LP        Do it again
 01E3 4C 00 01           JMP    DUMPT     Now--start SUPERTAPE
```

George W. Hawkins, NY

Here's a 2 task (foreground/background?) alternating scheduler routine. This routine (which resides in page one) divides the remainder of page one in half and manages two stacks while alternating control between each task. This allows two programs to be run together in the Kim as long as each program uses the stack or separate memory locations for the storage of temporary data. Set the address of task (program) one into 0100-01, and the address of task two into 0102-03. Connect A15 to E4 and start at 0107. Control will alternate as determined by the interval timer delay value and division rate in locations 0153 and 0155 respectively. Rescheduling will end when one of the programs issues a JMP START back to Kim.

```
****
0100  10           T1L   10.           TASK 1 START ADDRESS (currently = 0010)
0101  00           T1H   00.
0102  00           T2L   00.           TASK 2 START ADDRESS (currently = 0200)
0103  02           T2H   02.
0104  00           TSEL  00.           NEXT TASK TO EXECUTE (alternates)
0105  FF           TSTK  FF.           CURRENT STACK POINTER TASK 1
0106  A9           TST1  A9.           TASK 2

0107  A9 00        TINL  LDA I   00.   START WITH TASK 1
0109  8D 04 01           STA A   TSEL
010C  8D AD 01           STA A   01,AD ZERO TASK 2'S STATUS WORD
010F  A2 FF              LDX I   FF.   TASK 1 STACK POINTER
0111  8E 05 01           STX A   TSTK
0114  9A                 TXS           INIT STACK POINTER
0115  A9 A9              LDA I   A9.   TASK 2 STACK POINTER
0117  8D 06 01           STA A   TST1
011A  A9                 A9.           LOAD A
011B  39                  LOW    TINT  WITH INTERRUPT ADDRESS
011C  8D FE 17           STA A   IRQL
011F  A9                 A9.           LOAD A
0120  01                  HIGH   TINT
0121  8D FF 17           STA A   IRQH
0124  AD 02 01           LDA A   T2L   SET TASK 2 START ADDRESS
0127  8D AE 01           STA A   01,AE
012A  AD 03 01           LDA A   T2H
012D  8D AF 01           STA A   01,AF
0130  58                 CLI           INTERRUPTS ON
0131  A9 01              LDA I   01.   1 INTERVAL ON TIMER
0133  8D 0F 17           STA A   17,0F OF 1024
0136  6C 00 01           JMP @   T1L   START TASK 1

                                       TASK SWITCHING
0139  48           TINT  PHA           SAVE A
013A  8A                 TXA           SAVE X
013B  48                 PHA
013C  98                 TYA           SAVE Y
013D  48                 PHA
013E  BA                 TSX           GET STACK POINTER
013F  8A                 TXA
0140  AC 04 01           LDY A   TSEL  GET TASK SELECTOR
0143  99 05 01           STA AY  TSTK  SAVE ~~IT~~ STACK POINTER
0146  98                 TYA           SELECT OTHER TASK
0147  49 01              EOR I   01.
0149  A8                 TAY
014A  8D 04 01           STA A   TSEL
014D  B9 05 01           LDA AY  TSTK  START OTHER TASK
0150  AA                 TAX
0151  9A                 TXS           RESTORE STACK POINTER
0152  A9 01              LDA I   01.   RESCHEDULE 1 INTERVAL
0154  8D 0F 17           STA A   17,0F OF 1024
0157  68                 PLA
0158  A8                 TAY           RESTORE Y
0159  68                 PLA
015A  AA                 TAX           RESTORE X
015B  68                 PLA           RESTORE A
015C  40                 RTI           BACK TO MORE USEFUL THINGS
```

end

A CATALOG OF KIM-1 ROM BYTES. (Hal Gerden, Oakland, CA) The debug pregram TRACER by Larry Fish in the Aug. 1977 KILOBAUD makes innevative use ef the 6502 BIT instruction, using masks in memory lecations fer non-destructive testing ef bits in the accumulater. Since BIT lacks the immediate addressing mede, masks must be either at a zere-page er absolute address. Any byte in the KIM ROM can serve as a mask, te test net only single bits but alse the absence ef 2 er mere bits (e.g. BIT with a memory lecatien containing ØF will set the Z flag only if the accumulater bits Ø-3 are all Ø). With the help ef a simple pregram, I found 175 ef the 256 pessible bytes in the KIM ROM, and recorded the lowest address fer each ene. The table (high nybble en herizental, lew en vertical) gives this address (e.g., an Ø8 exiets at address 1981).

| | Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ø | 185D | 19A4 | 18Ø5 | 1974 | 1AØ9 | | 193F | | 1A69 | 183F | 198Ø | 1986 | | 181A | 1EFD | 1884 |
| 1 | 1C3Ø | 1C9D | 1F6B | 1C5F | 1897 | 1DF4 | 1825 | 1881 | | 198C | | 1CB4 | | 18BF | | 188A |
| 2 | 1853 | 1CA1 | 1E64 | 18Ø6 | 18ØB | 1FE2 | | 1887 | | | 1812 | | 1E1C | | | 1C15 |
| 3 | 19EE | 1CA5 | 19Ø5 | 186F | 181Ø | | 1CD4 | | | | | | 19CD | 1C7B | 1EF9 | 18C1 |
| 4 | 1855 | 19ØØ | 184Ø | | 19A9 | | 1813 | | 1CØF | | 1CBØ | | 198F | 1E68 | | 1C1Ø |
| 5 | 1E74 | 1C91 | 1F92 | | 1FE4 | | 1F92 | 1A94 | 185E | | 1CDC | | 1D2Ø | 1DF2 | | 1828 |
| 6 | 18BB | 1815 | 1CEF | | 1A47 | | 194E | | 1C11 | | 1DC8 | | 1E1Ø | | 1DAØ | 182E |
| 7 | 188D | 18Ø4 | 18Ø9 | | 19A2 | | | | 1FEE | | 19AC | | | | 1847 | 1837 |
| 8 | 1981 | 187Ø | | 1A58 | 19AØ | | 19C2 | 1E9C | 1994 | 195C | 194C | | 1A22 | 1E9B | 184D | 181B |
| 9 | 199F | 18Ø7 | 196F | | 1A66 | | 1957 | | | | 18ØØ | 1D2D | 18A5 | | 1894 | 1822 |
| A | 18AA | 1898 | 181D | | 1962 | 1C6B | 1C4D | 1817 | 1DØB | 1A7B | 1CF7 | 1C13 | 1819 | | 186C | 185F |
| B | | 1DE2 | | 1CEB | 1FDF | | | | | | | 1C93 | | 1C75 | 1996 | 1861 |
| C | 191C | 1864 | 19A1 | | 1862 | | 1C1C | | 197D | 18D6 | 199A | | 1C82 | | 18Ø3 | 1C39 |
| D | 189C | 1C63 | 1FØ9 | 1FDD | | | | | 18Ø2 | 1CF5 | 18Ø1 | 1E31 | 1836 | | 1834 | 1A52 |
| E | | 1AØ3 | 1A16 | | 1899 | 182B | | 19A7 | 197A | 1983 | 1997 | | 1EE2 | 1FF4 | 183A | 1C2Ø |
| F | 1871 | 1C73 | 1842 | 1E92 | 1863 | | 1967 | 1DEØ | | | 1C62 | 18ØE | 1FEA | 18B1 | 187E | 1892 |

## A Compiler for the 6502

Help is needed to complete development of a table driven compiler for the 6502. I have completed the parser and the production procedure programs but have had trouble in deciding which language to implement. Anyone interested in this compiler should contact me as to preference of language, desired features, etc.

I also need help in designing methods to implement parameter passing to subroutines, formatted I/O, and character string handling. If you feel that you could help solve these problems please write me and I will send more information.

I am currently on a SIMBOL compiler but I don't have a great deal of information on it. If anyone has access to BNF descriptions of this and other languages I would gladly pay for copying.

Contact: Ralph Deane, Box 33, Little Fort, B.C. Canada
VOE 2C0

## Program BRANCH

by Allen Anway
1219 North 21st St.
Superior, WI 54880

many times I've pressed the GO button and many times the KIM has flown off into hyperspace somewhere or the stack has punched out my carefully written program in page 1. In self defense I wrote BRANCH to go through my program, find the branch instructions and force the branch to see where I would end up. This program is fully relocatable and uses only locations 0000 and 0001 in the regular RAM. The program uses a few locations at the top of page 0, but this is all right as long as you do NOT single step BRANCH. Enter the program at the beginning and press the following buttons:

KEY 0 Decrement POINTH of address
KEY 1 Decrement POINTL of address
KEY 4 Increment POINTH of address
KEY 5 Increment POINTL of address

When keys held down continuously, the addresses will change continuously after a very short wait.

KEY C Seek branch instruction of the form %XXX1 0000 and stop there. (Be careful, program stops at DATA of this same form.)

KEY D Force the branch, starting at the branch instruction address.

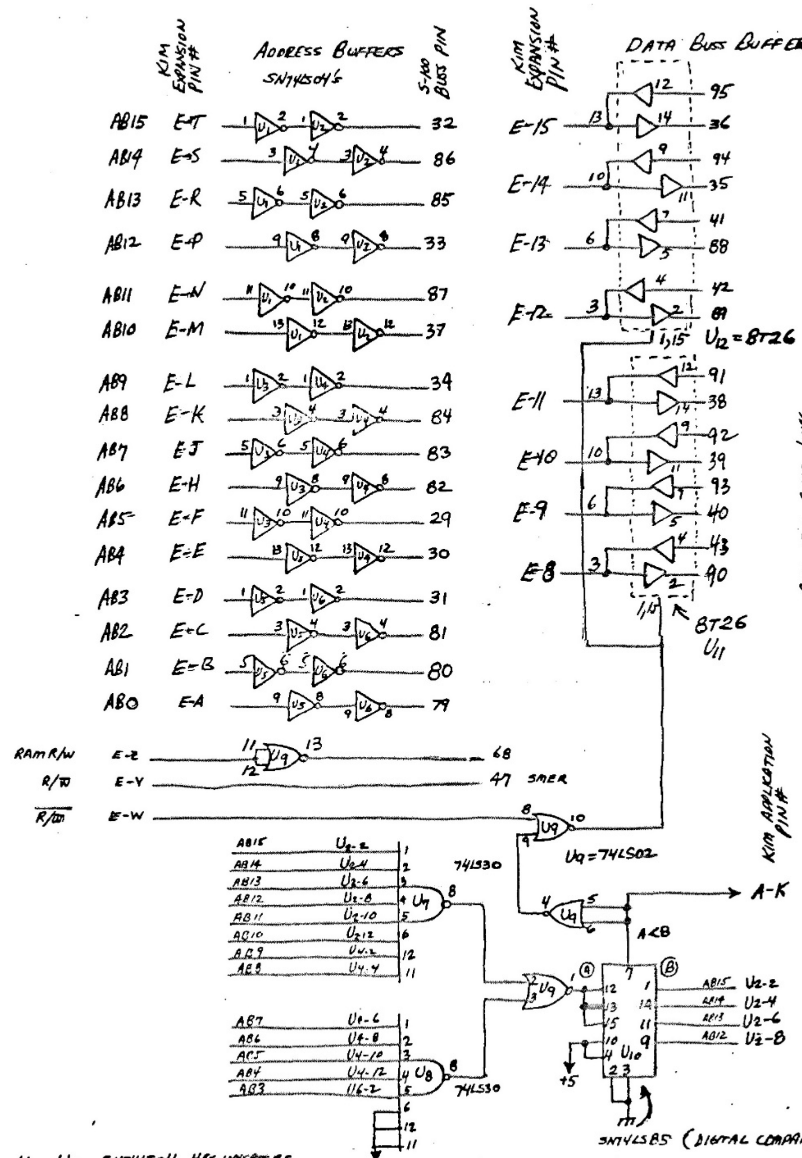
KEY E Above branched correctly, restore old branch address, remain in this program, next press C to look for another branch.

KEY F Above branched incorrectly, stop the program but restore the old branch address so you can correct the erroneous entry. Then press PC and GO and check your new entry by pressing D.

```
0343  08           STARTB  CLD
0344  A5 FA                LDA POINTL
0346  85 EF                STA PCL
0348  A5 FB                LDA POINTH
034A  85 F0                STA PCH   ; PC button is enabled
034C  A5 00                LDA TEML
034E  85 FA                STA POINTL
0350  A5 01                LDA TEMH
0352  85 FB                STA POINTH
- - - - - - - - - - - - - - - - - - -
0354  A9 80        A0      LDA #$80
0356  85 F3                STA NU    ; control repetition
0358  20 19 1F  A1         JSR SCAND
035B  F0 F7                BEQ A0    ; A0 on no key pressed
035D  20 6A 1F             JSR GETKEY
0360  85 F4                STA KEY
0362  A5 F3                LDA NU
0364  85 F1                STA NUM
0366  20 19 1F  A2         JSR SCAND
0369  F0 08                BEQ A3    ; A3 on key released
036B  C6 F1                DEC NUM
036D  D0 F7                BNE A2    ; A2 on key depressed short time
036F  A9 10                LDA #$10  ; key held long time,
0371  85 F3                STA NU    ;   go for repetition
- - - - - - - - - - - - - - - - - - -
0373  A5 F4        A3      LDA KEY
0375  C9 0F                CMP #$0F
0377  D0 08                BNE A4    ; A4 on not key F
0379  A5 00                LDA TEML  ; key F = leave program
037B  85 FA                STA POINTL;   but set up for old branch instruc.
037D  A5 01                LDA TEMH
037F  85 FB                STA POINTH
0381  4C 4F 1C             JMP START
- - - - - - - - - - - - - - - - - - -
0384  C9 0C        A4      CMP #$0C
0386  D0 10                BNE A5    ; A5 on not key C
0388  20 63 1F  A41        JSR INCPT ; key C = seek branch
038B  20 19 1F             JSR SCAND ; pick up program step from SCAND
038E  A5 F9                LDA INH
0390  29 1F                AND #$1F  ; look for branch format
0392  C9 10                CMP #$10
0394  D0 F2                BNE A41   ; A41 on branch not found
0396  F0 BC                BEQ A0    ; stop looking, branch found
- - - - - - - - - - - - - - - - - - -
```

```
0398  C9 0D      A5      CMP #$0D
039A  D0 3A              BNE A8     ; A8 on not key D
039C  A5 FA              LDA POINTL; key D = perform jump
039E  85 00              STA TEML
03A0  A5 FB              LDA POINTH
03A2  85 01              STA TEMH
03A4  20 63 1F           JSR INCPT ; go to next location
03A7  20 19 1F           JSR SCAND ; pick up branch distance
03AA  A5 F9              LDA INH   ;   from INH
03AC  48                 PHA
03AD  20 63 1F           JSR INCPT ; next location for easy calc.
03B0  68                 PLA
03B1  18                 CLC
03B2  10 09              BPL A52   ; A52 on branch forward
03B4  65 FA              ADC POINTL; branch backward
03B6  B0 02              BCS A51   ; A51 on no page crossed
03B8  C6 FB              DEC POINTH; page crossed backward
03BA  18         A51     CLC
03BB  90 06              BCC A53
03BD  65 FA      A52     ADC POINTL
03BF  90 02              BCC A53   ; A53 on no page crossed
03C1  E6 FB              INC POINTH; page crossed forward
03C3  85 FA      A53     STA POINTL
03C5  18                 CLC
03C6  90 8C              BCC A0    ; end of calculation
- - - - - - - - - - - - - - - - - - -
03C8  C6 FB      A6      DEC POINTH; from A7 and A8
03CA  B0 8C      A61     BCS A1    ; absolute jump
- - - - - - - - - - - - - - - - - - -
03CC  C6 FA      A7      DEC POINTL; from A8
03CE  A5 FA              LDA POINTL
03D0  C9 FF              CMP #$FF
03D2  F0 F4              BEQ A6
03D4  90 82      A71     BCC A1    ; absolute jump
- - - - - - - - - - - - - - - - - - -
03D6  C9 00      A8      CMP #$00  ; examine remaining keys
03D8  F0 EE              BEQ A6
03DA  C9 01              CMP #$01
03DC  F0 EE              BEQ A7
03DE  C9 04              CMP #$04
03E0  F0 0B              BEQ A9
03E2  C9 05              CMP #$05
03E4  F0 0B              BEQ A10
03E6  C9 0E              CMP #$0E
03E8  F0 0C              BEQ A11
03EA  18                 CLC
03EB  90 E7              BCC A71   ; A71 on no legal key pressed
- - - - - - - - - - - - - - - - - - -
03ED  E6 FB      A9      INC POINTH
03EF  B0 D9              BCS A61   ; absolute jump
- - - - - - - - - - - - - - - - - - -
03F1  20 63 1F   A10     JSR INCPT
03F4  B0 D4              BCS A61   ; absolute jump
- - - - - - - - - - - - - - - - - - -
03F6  A5 00      A11     LOA TEML  ; key E = pick up old branch
03F8  85 FA              STA POINTL;   but remain in program
03FA  A5 01              LDA TEMH
03FC  85 FB              STA POINTH
03FE  B0 CA              BCS A61   ; absolute jump
```

end



U1 - U6 SN74LS04 Hex Inverters
U7 - U8 SN74LS30 8 Input Nand Gates
U9 SN74LS02 Quad Nor Gate
U10 SN74LS85 Binary Comparator
U11, U12 8T26 Quad Tristate Transciever

Don't forget to add enough bypass caps.

Note: Line A-K goes low when address is less than 2000 Hex or more than FFF8 hex.

Kim to S-100 Buss Adapter:
By Jim Pollock
7-1-77

Also Tie S-100 Lines #45, #46, and #70 to ground.

Jim also recommends the Ithaca Audio 8K Ram board (125.00)

Ithaca Audio
P.O. Box 91
Ithaca, N.Y. 14850

# Kim-1 → S-100 Bus Adapter

Jim Pollock
New Egypt, N.J.

(Now you can take advantage of all that low-cost memory)

A N ER OF YOU HAVE REQUESTED BASIC LEVEL PROGRAM EXPLANATIONS

HARVEY LAYS AN EXCELLENT TUTORIAL ON US..
- ERIC -

## A SIMPLE MUSIC PROGRAM FOR KIM by Harvey Heinz

Undoubtedly, the single most popular use for hobby computers is the programming and playing of games. However, another common use is the playing of music with the micro-computer. Most programs used for this purpose tend to be quite elementary and so it follows that the music generated leaves much to be desired from a quality point of view. Dispite this, music is a good subject for the computer hobbyist to pursue, for the following reasons.

1. The basic principals are very simple but can be elaborated on to any degree desired. In fact,electronic music can become a hobby in itself.
2. Writing a music program makes one very consious of execution times of his machines instruction set.
3. Playing music on the computer is ideal for demonstrating to the layman the versatality of these machines.

As a KIM-1 owner, I had an additional reason for attempting to write such a program. As you know, the 6530 has a programmable interval timer that may be used to interrupt the MPU. I felt that by using this feature, a very simple program could be designed. At the same time I would be gaining experience in using this valuable feature, and also learn something about using the interrupt.

The program which evolved is flow-charted in Fig. 1. Actually there are two separate programs. The main routine consists mostly of initialization. The working part of this program though is the timing loop at the end. Every 4 microseconds Reg. Y is decramented. When the contents of this register become 0, the output is toggled, thus pusing the speaker to the opposite position to the one preveously held. Register Y is then re-initialized, and the process repeats. This will happen continuously until the IRQ line is triggered by the interrupt. The value Reg. Y is initialized to determines the frequency of the note being played.

The interrupt routine is only a little more complicated. The timer has originally been initialized to a value called TEMPO. This value is what determines whether the tune plays fast or slow. The timer is loaded with this value by accessing it with address 170F. This automatically programs the timer to count down 1 for every 1024 clock periods. At the same time, PB7 is initialized to act as an interrupt flag.

Approximately 20 times per second (with TEMPO equal to 28$_{16}$) the timer will reach 0 and initiate an interrupt. The constant LENGTH is then decremented and tested for 0. If not 0, the timer is re-initialized, and return is then made to the main program. If LENGTH is equal to 0, the interrupt fetches the next note and next duration from the tune table after first checking that the tune is not over. After re-initializing the timer, return is made to the main routine which will now generate the new note.

If the end of tune has been reached during the interrupt, a jump is made direct to the monitor, thus stopping the program. While this is not the proper way to return from an interrupt, in this case it does no harm. Fig. 2 is a listing of both programs.

The tune is listed as a separate table (from the program) and so may be easily changed. Fig. 3 is a listing for the verse and chorus of Swanee River. Even bytes are constants which represent the frequency of the note. The following odd byte is a constant which represents the duration of the note. Refer to Fig. 4 for the correct values to use when coding a different tune.

A suitable value should be stored in TEMPO (00EA) to determine the speed the tune is played at. Try varying this value for interesting effects. The first empty address after the table should be stored at 00EB to stop the program when the tune is over.

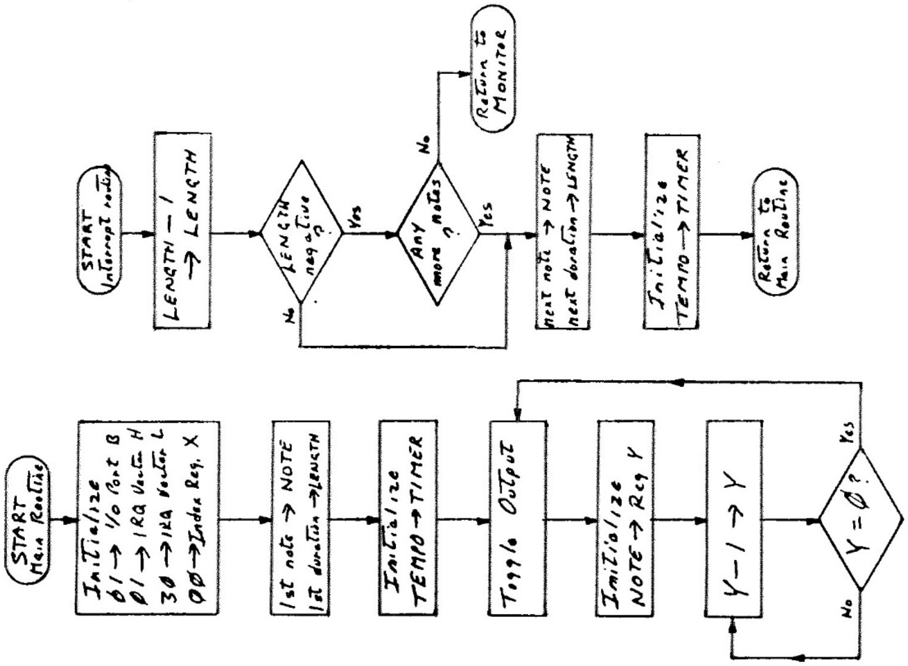
Fig. 4 is a list of musical notes with their correct frequency and period in microseconds. Because our demonstration program has only a single time delay loop, the period must be divided by 4 to make it less then 1024. This does no harm except to raise the frequency generated. Our computer now sounds like a picolo or flute. This modified period is again divided by 4 (our 4 μsec. timing loop) to give the proper argument for that frequency. As this number is decimal,it is finally converted to Hexadecimal to give the correct constant for that note.

The duration argument is derived by determining the shortest note in the selected musical piece. Assign an arbitrary value for this duration. Then simply assign integer multiples of this value for the longer notes. For Swanee River, I used 05 to represent 1 beat. Combining this value with 27 or 28 for TEMPO works out about right.

The hardware end of the project is also simple. Refer to page 57 of your User Manual. Hook up the speaker and transister amplifier as per the diagram, but connect it to PBO (A9). Then connect PB7 (A15) to IRQ (E4). This last connection should be made through a switch or alligator clip so it can be broken when using the cassette interface.

Using the program can be a lot of fun, as well as being educational. Try slowing down or speeding up the music by changing just the 1 value TEMPO. That's a range of 256 to 1. Or play the tune backwards by changing only a few bytes in the program (decrement X). Or don't load a table at all.Just use the random numbers in memory as a computer generated tune. Anyway have fun. Isn't that what hobby computers are all about?

Fig. 1.. MUSIC PROGRAM



more...

Fig. 2--Music Program for KIM-1

A. Main Routine

```
A9  01        0100         LDA #01        Initialize
8D  03  17       2         STA PBDD            I/O Port B
8D  FF  17       5         STA 17FF            IRQ Vector High
A9  27           8         LDA #27             IRQ Vector low
8D  FE  17       A         STA 17FE
A2  00           D         LDX #00             Register X
B5  00           F         LDA TABLE,X
85  E8        0111         STA NOTE       Store first note in NOTE
E8               3         INX
B5  00           4         LDA TABLE,X
85  E9           6         STA LENGTH                and LENGTH
A5  EA           8         LDA TEMPO      Initialize TIMER
8D  0F  17       A         STA TIMER
EE  02  17       D  PLAY   INC PBO        Toggle output
A4  E8        0120         LDY NOTE       Initialize Reg. Y to NOTE
88               2  DELAY  DEY            Decrement Reg. Y
D0  FD           3         BNE DELAY      If not zero, return
F0  F6        0125         BEQ PLAY       Time delay complete
```

B.Interrupt Routine

```
C6  E9        0127         DEC LENGTH     Decrement LENGTH
30  06           9         BMI NEXTN      If zero, get next note
A5  EA           B         LDA TEMPO      Reinitialize TIMER
8D  0F  17       D         STA TIMER
40            0130         RTI            And return to main routine
E8               1  NEXTN  INX            Increment Index Register
E4  EB           2         CPX END        Test for tune over
D0  03           4         BNE CONT       No? then continue
4C  4F  1C       6         JMP START      Yes. Go to KIM monitor
B5  00           9  CONT   LDA TABLE,X    Fetch next note (Freq.)
85  E8           B         STA NOTE       and store in NOTE
E8               D         INX            Increment Index Reg.
B5  00           E         LDA TABLE,X    Fetch next duration
85  E9        0140         STA LENGTH     and store in LENGTH
A5  EA           2         LDA TEMPO      Reinitialize TIMER
8D  0F  17       4         STA TIMER
40            0147         RTI            Return to main routine
```

```
0000  Start of TABLE                          TABLE
00E8  Location of current note frequency      NOTE
00E9  Location of current note duration       LENGTH
00EA  Constant here determines speed of tune  TEMPO
00EB  Contains first empty address after tune END
```

THE FIRST BOOK OF KIM is becoming available in stores across the country. Stan Ockers, Jim Butterfield, and your editor put this book together with the idea of helping newcomers to our hobby to get up to speed on the KIM. (Of course, the book's not just applicable to newcomers). The book includes a beginners guide to programming, several tutorials on hooking things up to KIM, and a large number of game and utility type programs. (many of which have not been published as yet). The First Book Of KIM is 180 pages long in an 8½ X 11 format. It is available for $9.00 (plus $.50 postage) from: ORB, P.O. Box 311, Argonne, Ill. 60439. Personal checks will have to clear the bank, so please send a cashiers check or money order in U.S. funds. Ill. residents please add sales tax.

•••••••••••••••••••••••••••

Fig.3-Table For Swanee River Tune

| | | | | |
|---|---|---|---|---|
| E | 4 | 0000 | BE | 14 |
| D | 1 | 2 | D5 | 05 |
| C | 1 | 4 | EF | 05 |
| E | 1 | 6 | BE | 05 |
| D | 1 | 8 | D5 | 05 |
| C | 2 | A | EF | OA |
| C | 2 | C | 77 | OA |
| A | 1 | E | 8E | 05 |
| C | 3 | 0010 | 77 | OF |
| G | 4 | 2 | 9F | 14 |
| E | 2 | 4 | BE | OA |
| C | 2 | 6 | EF | OA |
| D | 8 | 8 | D5 | 28 |
| E | 4 | A | BE | 14 |
| D | 1 | C | D5 | 05 |
| C | 1 | E | EF | 05 |
| E | 1 | 0020 | BE | 05 |
| D | 1 | 2 | D5 | 05 |
| C | 2 | 4 | EF | OA |
| C | 2 | 6 | 77 | OA |
| A | 1 | 8 | 8E | 05 |
| C | 3 | A | 77 | OF |
| G | 2 | C | 9F | OA |
| E | 1 | E | BE | 05 |
| C | 1 | 0030 | EF | 05 |
| D | 4 | 2 | D5 | 14 |
| C | 8 | 4 | EF | 28 |
| B | 3 | 0036 | 7F | OF |
| C | 1 | 8 | 77 | 05 |
| D | 2 | A | 6A | OA |
| G | 5 | C | 9F | 19 |
| A | 1 | E | 8E | 05 |
| G | 2 | 0040 | 9F | OA |
| C | 4 | 2 | 77 | 14 |
| A | 2 | 4 | 8E | OA |
| F | 2 | 6 | B3 | OA |
| A | 2 | 8 | 8E | OA |
| G | 8 | A | 9F | 28 |
| E | 4 | C | BE | 14 |
| D | 1 | E | D5 | 05 |
| C | 1 | 0050 | EF | 05 |
| E | 1 | 2 | BE | 05 |
| D | 1 | 4 | D5 | 05 |
| C | 2 | 6 | EF | OA |
| C | 2 | 8 | 77 | OA |
| A | 1 | A | 8E | 05 |
| C | 3 | C | 77 | OF |
| G | 2 | E | 9F | OA |
| E | 1 | 0060 | BE | 05 |
| C | 1 | 2 | EF | 05 |
| D | 4 | 4 | D4 | 14 |
| C | 7 | 6 | EF | 23 |

Load OOEB (END) with 68

Load OOEA (TEMPO) with 28

Fig. 4--- Musical Notes with Frequency,Period, & Argument

| Note | Frequency | Period | Period/4 | Constant<br>Dec. | Hex. |
|---|---|---|---|---|---|
| C | 261.62 | 3822.3 | 956 | 239 | EF |
| C# | 277 | 3608 | 902 | 226 | E2 |
| D | 294 | 3405 | 851 | 213 | D5 |
| D# | 311 | 3214 | 804 | 201 | C9 |
| E | 329.63 | 3033.8 | 759 | 190 | BE |
| F | 349 | 2864 | 716 | 179 | B3 |
| F# | 370 | 2703 | 676 | 169 | A9 |
| G | 392 | 2551 | 638 | 160 | AO |
| G# | 415 | 2408 | 602 | 151 | 97 |
| A | 440 | 2273 | 568 | 142 | 8E |
| A# | 466 | 2145 | 536 | 134 | 86 |
| B | 493 | 2025 | 506 | 127 | 7F |
| C | 523 | 1911 | 478 | 120 | 78 |
| C# | 554 | 1804 | 451 | 113 | 71 |
| D | 587 | 1703 | 426 | 107 | 6B |
| D# | 622 | 1607 | 402 | 101 | 65 |
| E | 659 | 1517 | 379 | 95 | 5F |
| F | 698 | 1432 | 358 | 90 | 5A |
| F# | 740 | 1351 | 338 | 85 | 55 |
| G | 784 | 1276 | 319 | 80 | 50 |
| G# | 831 | 1204 | 301 | 75 | 4B |
| A | 880 | 1136 | 284 | 71 | 47 |
| A# | 932 | 1073 | 268 | 67 | 43 |
| B | 988 | 1012 | 253 | 63 | 3F |
| C | 1047 | 956 | 239 | 60 | 3C |

# AN A/D CONVERTER. FROM... WILL HAPGOOD
WALTHAM, MASS

Here is a circuit for making very accurate A/D conversions using a Motorola dual-slope conversion chip. With the values shown, I get conversions of up to 1400 counts with 1 bit accuracy compared to the best digital voltmeter we have; zero drift is non measurable. With a larger integrating capacitor, the circuit will count past 2000 counts; with a longer software timing constant, you can get a full 16 bit count, but with a longer conversion time than the approximately 50 msec. my program uses.

The input signal must be positive, although you can float the return line by about a volt if desired. I set the two potentiometers to mid-scale before beginning adjustments so they won't be too far off. The transistor can be any PNP device, and is for protection against reversed input polarity, which otherwise might latch up the chip. Finally, avoid snapping the power supply on(by inserting a chip into a live socket); it can make the chip very non-linear, or even dead.

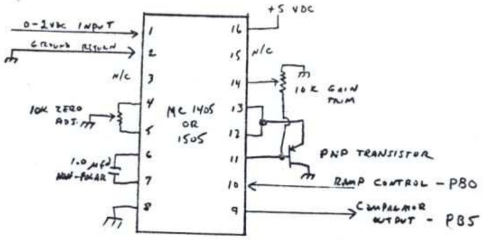
The software is relocatable. It is written for the output line to be PB0 in KIM, and the input line to be PB5. The program controls the ramp line; when it is on, the 1405 integrator is going negative. When it goes below zero(actually below a reference voltage), the ramp is reset and the integrator starts going positive. The up-ramp is timed once it crosses zero. At the end of the timed up ramp, the ramp control line is set, and the time required for the integrator to reach zero is counted. This is proportional to the input value. Subtracting an offset of 5 or 10 percent of the upramp count improves operation near zero; the exact amount subtracted is not critical. Notice the instructions to disable interrupts during the critical counting periods; the software must not be disturbed during this period.

The spec sheet on the MC1505L and Motorola Application Note #AN-757 contain more information on the chip and its use. I am currently using this circuit preceeded by an analog multiplexer to read up to 16 inputs accurately in less than 1 second, using only two computer interface lines. I find the circuit much easier to use than a 12 bit parallel A/D, and much cheaper in the bargin.

The chip operates by integrating a current proportional to the input for a fixed time period(set by the timing constant for the up-ramp). Then a down ramp period subtracts a reference current until the integrating capacitor returns to zero. Thus many circuit variables balance out. Loading Y with $06 and X with $00 is am up-ramp constant of $0600, or 1500 decimal. During the up-ramp, this number is counted to zero to give the up ramp delay time. Once $00 is reached, the ramp direction is reversed, and the same registers are counted up until the integrating capacitor returns to its original level. With the software as it is, I get 1500 decimal counts at an input voltage of 1.5 volts. However, the circuit counts somewhat higher than this before getting non-linear.

To reach a full 16 bit count of 65,ooo, a larger up ramp timing constant can be specified. This will charge the timing capacitor for a longer time, and result in higher counts for a particular input voltage. You ~~May~~ will have to increase the size of the integrating capacitor to prevent it from limiting; and conversions will take longer as the size of the count goes up. The software as shown results in a 16 bit result but with a maximum count of 2000 decimal or so(an 11 bit range). Fiddle with the timing constant until the system counts linearly up to the desired range; then set the zero offset constant to between 5% and 10% of the up-ramp constant. Adjust the zero offset constant until the circuit zeros; then trim the gain potentiometer for the exact gain required, and finally, re-trim the zero with the zero control.

I've included another listing which adds a simple but clear binary to bcd conversion. The references to 16 bit numbers should probably be changed to 12 to avoid confusion.



MC1405- A/D CIRCUIT

```
; INPUT MODULE OPERATES A SET-RESET DUAL-SLOPE A-D CONVERTER.
; INPUT LINE = PB5 ($20)
; PB0 IS OUTPUT LINE TO A-D.
; THIS MODULE INCLUDES BCD CONVERSION.
; INPUTS = NONE.  OUTPUTS = MSD IN X; LSD IN Y.
.SKIP 2
INPUT   LDA #<00000001 TURN RAMP ON AT PB0
        ORA PBDATA
        STA PBDATA
        LDA #$20       MASK FOR THIS INPUT
TEM1    BIT PBDATA
        BNE TEM1       LOOP TILL COMP GOES LOW
        LDX #0
        LDY #$06       TIMING CONSTANT FOR UP-RAMP
        DEC PBDATA     TURN RAMP OFF
TEM2    BIT PBDATA
        BEQ TEM2       LOOP TILL COMP GOES HIGH
        SEI           DISABLE IRQ
TEM3    DEX
        BNE TEM3
        DEY
        BNE TEM3
        INC PBDATA     TURN RAMP ON
TEM4    INX
        BNE TEM5
        INY
TEM5    BIT PBDATA
        BNE TEM4
        CLI            ENABLE IRQ
        DEC PBDATA     LEAVE RAMP OFF TO EQUALIZE CONVERSION TIMES
        TXA            SUBTRACT OFFSET TO IMPROVE OPERATION NEAR ZERO.
        SEC
        SBC #$40
        TAX
        TYA
        SBC #0
        TAY
;                     AT THIS POINT, 16 BIT BINARY IS IN Y AND X.
```

more...

```
.SKIP 4
; SUB-MODULE BCD. NORMALLY ENTERED FROM INPUT ABOVE, BUT
; CAN ALSO BE CALLED INDEPENDENTLY.
;
; THIS MODULE CONVERTS A 16 BIT BINARY NUMBER INPUTTED IN
; Y AND X INTO THE 4 DECIMAL DIGITS CONTAINED BY MSD AND LSD.
; IT COUNTS DOWN Y, ADDING 256 TO LSD,MSD; THEN IT COUNTS DOWN
; X WHILE ADDING 1.
.SKIP 1
BCD     SED             USE DECIMAL ADDITION
        LDA #0          CLEAR OUTPUTS
        STA LSD
        STA MSD
        CPY #0          IF MSBITS = 0, DO LSBITS
        BEQ BCD2
BCD1    CLC             ADD 256 TO OUTPUT
        LDA LSD
        ADC #$56
        STA LSD
        LDA MSD
        ADC #2
        STA MSD
        DEY             AND DECREMENT MSBITS BY 1
        BNE BCD1        LOOP TILL ZERO
.SKIP 1
BCD2    CPX #0          IF LSBITS = 0, DONE
        BEQ BCD4
BCD3    CLC             ADD 1 TO OUTPUT
        LDA LSD
        ADC #1
        STA LSD
        LDA MSD
        ADC #0
        STA MSD
        DEX             AND DECREMENT LSBITS
        BNE BCD3        LOOP TILL ZERO
BCD4    LDX MSD
        LDY LSD
        CLD
        RTS
 COPY COMPLETE.
```

KIM BLACKJACK
May 28, 1977

Jim Butterfield
14 Brooklyn Avenue
Toronto M4M 2X5, Canada

**Description:**

KIM uses a 'real' deck of cards in this game. So when you've seen four aces going by, you know that there will be no more - until the next shuffle.

BLACKJACK starts at address 0200. You'll see the cards being shuffled - the word SHUFFL appears on the display - and then KIM will ask how much you want to bet.

You'll start with an initial amount of $20. Your balance is always shown to the right of the BET? question, so on the first hand, you'll see BET? 20 on the display.

You may bet from $1 to $9, which is the house limit. The instant you hit key 1 to 9 to signal your bet, KIM will deal. Of course, you can't bet more money than you have ... and KIM ignores freeloaders who try to bet a zero amount.

After the deal, you'll see both your cards on the left of the display, and one of KIM's cards on the right. (KIM's other card is a "hole" card, and you won't see it until it's KIM's turn to play). Aces are shown as letter A, face cards and tens as letter F, and other cards as their value, two to nine. As always, Aces count value 1 or 11 and face cards count 10.

You can call for a third card by hitting the 3 button .. then the fourth card with the 4 button, and so on. If your total goes over 21 points, KIM will ungrammatically say BUSTED, and you'll lose. If you get five cards without exceeding 21 points, you'll win automatically. If you don't want any more cards, hit key 0. KIM will report your point total, and then will show and play its own hand. KIM, too, might go BUSTED or win on a five-card hand. Otherwise, the most points wins.

From time to time, KIM will advise SHUFFL when the cards start to run low.

Remember that you have a good chance to beat KIM at this game. Keep track of the cards that have been dealt (especially aces and face cards), and you're likely to be a winner!

KIM BLACKJACK

```
0200 A2 33    START  LDX #51      52 cards in deck
0202 8A       DK1    TXA          Create deck
0203 95 40           STA DECK,X    by inserting cards
0205 CA              DEX             into deck
0206 10 FA           BPL DK1          in sequence
0208 A2 02           LDX #2       Set up 3 locations
020A BD BB 03 INLOP  LDA INIT,X      ..into..
020D 95 75           STA PARAM         zero page
020F CA              DEX          addresshi/ dpt/ amt
0210 10 F8           BPL INLOP
0212 AD 04 17        LDA TIMER    use random timer
0215 85 80           STA RND       to seed random chain
0217 D8       DEAL   CLD          main loop repeats here
0218 A6 76           LDX DPT      next-card pointer
021A E0 09           CPX #9       less than 9 cards?
021C B0 34           BCS NOSHUP   9 or more, don't shuffl
               ; shuffle deck
021E A0 D8           LDY #SHUF-$300  Set up SHUFFL msg
0220 20 57 03        JSR FILL           put in WINDOW
0223 A0 33           LDY #51           ripple 52 cards
0225 84 76           STY DPT      set full deck
0227 20 30 03 SHLP   JSR LIGHT    illuminate display
022A 38              SEC
022B A5 81           LDA RND+1    Generate
022D 65 82           ADC RND+2     new
022F 65 85           ADC RND+5      random
0231 85 80           STA RND         number
0233 A2 04           LDX #4
0235 B5 80    RMOV   LDA RND,X    move over
0237 95 81           STA RND+1,X   the random
0239 CA              DEX             seed numbers
023A 10 F9           BPL RMOV
023C 29 3F           AND #$3F     Strip to 0-63 range
023E C9 34           CMP #52      Over 51?
0240 B0 E5           BCS SHLP     yes, try new number
               ; swap each card into random slot
0242 AA              TAX
0243 B9 40 00        LDA DECK,Y   get next card
0246 48              PHA          save it
0247 B5 40           LDA DECK,X   get random card
0249 99 40 00        STA DECK,Y    into position N
024C 68              PLA          and the original card
024D 95 40           STA DECK,X    into the random slot
024F 88              DEY          next in sequence
0250 10 D5           BPL SHLP     bck for next card
```

more ↴

# 'XIM'

(Extended I/O Monitor)

A TTY, command oriented, programming tool for KIM-1

1. Resides in 1K of memory. Relocatable (with checklist) and ROM-able.

2. Adds 17 commands to resident KIM TTY monitor.

3. Includes 4 user defined commands for expansion.

4. Designed around a modular concept for easy modification.

## FUNCTIONS

*Load alpha-numeric (ASCII) characters into ram via TTY.
*Print a memory block on the TTY as alpha-numeric (ASCII) characters.
*Calculate relative branches.
*Compare two data blocks and display all discrepancies.
*Load op-codes and operands into memory sequentially via TTY.
*Execute a program at a designated address.
*HEX Dump: Display memory as a 16 column matrix of two digit HEX codes.
*Jump to the KIM monitor.
*Fill a data block with a constant.
*Move one block of data to another.
*Block-search for a string of data up to 256 bytes long in any given block and display the starting address(es) of the string.
*Set up the audio tape address buffers via TTY in sequential fashion.
*CONTROL D. Used for command termination, during initialization.

### Break point (BRK) service routine.

BRK point processing routine saves and displays all CPU registers on the TTY. Status register is printed as a string of 1's and 0's for program debugging.

Features OP-code reinsertion at BRK point for multi BRK processing.

-------------

Manual & Cassette: $12.00
Manual & Punched tape: $10.00
(post paid USA)
NJ residents add 5% tax.

PYRAMID DATA SYSTEMS
6 Terrace Ave.
New Egypt, N.J.
08533

A NUMBER OF YOU HAVE WANTED ALIST OF
KIM MONITOR ROUTINES WITH EXPLANATIONS

B. STRANDTOFT 03.02.77
Mollebakken 27
GUDERUP
6430 NORDBURG
DENMARK

```
**** KIM-1 RESIDENT PROGRAM'S AND SUBROUTINE'S ****

-NAME-   -COMMENT-

MAIN
      DUMPT    DUMP MEM TO TAPE
      LOADT    LOAD MEM FROM TAPE
      INTVEB   SUB TO MOVE SA TO VEB +1,2
      CHKT     COMPUTE CHKSUM FOR TAPELOAD. RTN USES Y TO SAVX +
      OUTBTC   OUTPUT ONE BYTE. USES Y TO SAVX BYTE
      OUTBT    OUTBTC WITHOUT CHKSUM
      HEXOUT   CONVERT LSD OF A TO ASCII AND OUTPUT TO TAPE
      OUTCHT   OUTPUT TO TAPE ONE ASCII CHAR VIA SUB'S ONE + ZERO
SUB'S
      ONE      OUTPUT '1' TO TAPE. 9 PULSES 138 MICROSEC EACH
      ZRO      OUTPUT '0' TO TAPE. 6 PULSES 207 MICROSEC EACH
      INCVEB   SUB TO INC VEB+1,2
      RDBYT    SUB TO READ BYTE FROM TAPE
      RDBYT2   MULTI ENTRY POINT
      PACKT    PACK A=ASCII INTO SAVX AS HEX DATA
      RDCHT    GET 1 CHAR FROM TAPE. RETURN CHAR IN A. USE SAVX+1 TO ASM CHAR
      RDBIT    GETS ONE BIT FROM TAPE AND RETURNS IT IN SIGN OF A
MAIN
      PLLCAL   OUTPUT 166 MICROSEC PULSE STRING FOR TAPE-PLL CALIBRATION
      SAVE     KIM ENTRY VIA STOP (NMI) OR BRK (IRQ)
      SAVE1    KIM ENTRY VIA JSR (A LOST)
      SAVE2    (ISS) X, Y, S
      RST      KIM ENTRY VIA RST
      DETCPS   DETECT CHAR PER SEC ( BAUD-RATE )
      START    MAKE TTY/KB SELECTION
      CLEAR    CLEAR INPUT BUFFER INH, INL AND READ
      READ     GET CHAR
      TTYKB    MAIN ROUTINE FOR KEYBOARD AND DISPLAY. IF NO KEY, A= 0
      GETK     KIM-KEYBOARD FETCH-PROGRAM
      GET5     TEST CHAR IN DETCPS
      DATA     SHIFT CHAR IN A INTO HIGH ORDER NIBBLE AND DISPLAY
      ADDR     DISP ADR
      STEP     INCPT + START
      PCCMD    DISPLAY PC BY MOVING PC TO POINT
      LOAD     LOAD PAPERTAPE FROM TTY. CHECK FOR ';'
      LOADS    LOAD PAPERTAPE FROM TTY. CHECK FOR BYTECOUNT
      DUMP     DUMP TO TTY FROM OPEN CELL ADRESS TO LIMHL, LIMHH
      SPACE    OPEN NEW CELL
      SHOW     PRINT OPEN CELL
      RTRN     OPEN NEXT CELL
      GOEXEC   RUN-ISS. PROGRAM RUNS FROM OPEN CELL ADR
      SCAN     TTY-CMD DETECTION PROG
      FEED     OPEN PREVIOUS CELL. PRINT
      MODIFY   GET CONTENTS OF INPUT BUFF INL AND STORE IN LOC SPECIFIED BY POINT
SUB'S
      PRTPNT   SUB TO PRINT POINTL, POINTH
      CRLF     SUB TO PRINT CR + LF
      PRTST    PRINT STRING OF ASCII CHAR FROM TOP+X TO TOP
      PRTBYT   PRINT ONE HEX BYTE AS TWO ASCII CHAR'S
      HEXTA    CONVERT TO HEX NIBBLE AND PRINT ASCII
      GETCH    GET 1 CHAR FROM TTY. CHAR IN A. X PRESERVED. Y = FF
      GET5     GETCH MULTI ENTRY POINT
      INITS    INITIALIZATION FOR SIGMA
      INIT1    INITS MULTI ENTRY POINT
      OUTSP    PRINTS 1 SPACE
      OUTCH    PRINT 1 CHAR = A. X PRESERVED. Y = FF
      DELAY    DELAY 1 BIT. TIME AS DETERMED BY DETCPS
      DEHALF   DELAY HALF BIT TIME
      AK       KEY NOT DEP OR TTY MODE, A=0. KEY DEP OR KB MODE, A NOT ZERO
```

```
ONEKEY   LIKE AK, BUT X, Y NOT INITIATED
SCAND    OUTPUT 3 BYTES TO 7 SEGMENT DISPLAY. DATA SPECIFIED BY POINT
SCANDS   OUTPUT TO 7 SEGMENT DISPLAY.
CONVD    CONVERT AND DISP HEX. (SCAND)
INCPT    SUB TO INCREMENT POINTL, POINTH
GETKEY   FROM KEYBOARD. A = KEYVALUE. ILLEGAL OR NO KEY FOR A GT. 15
CHK      SUB TO COMPUTE CHECK SUM
GETBYT   GET 2 HEX CHAR'S AND PACK INTO INL, INH. X PRESERVED. Y = 0
PACK     SHIFT CHAR IN A INTO INL, INH. A = 0 FOR HEX
HEXNUM   CONVERT TO HEX NUM WITHOUT CHECK. A = 0
HEXALP   CONVERT TO HEX ALPHA
UPDATE   SHIFT A INTO MSD AND STORE IN I/O BUFFER INL, INH
OPEN     MOVE I/O BUFFER INL, INH TO POINTL, POINTH

TAB      KIM MESSAGE TABLE AND 7-SEGMENT CONVERT TABLE
```

## A KIM BIBLIOGRAPHY FROM....

WILLIAM R. DIAL
438 ROSLYN AVE
AKRON, OHIO
44320

Ohio Scientific Instruments, 11679 Hayden Ave., Hiram, OH 44234
"Model 300 Computer - Trainer Lab Manual"
A series of 20 programs for instruction on the 6502 microprocessor based Model 300 Trainer. Programs are easily adapted to KIM-1 operation.

Ohio Scientific Instruments, 11679 Hayden St., Hiram, OH 44234
"Application Note No. 2"
OSI 480 Backplane and Expansion System.

Ohio Scientific Instruments, 11679 Hayden St., Hiram, OH 44234
"OSI Application Note No. 5"
Interfacing OSI Boards to other systems including KIM-1.

Ohio Scientific Instruments, 11679 Hayden St., Hiram, OH 44234
"OSI Model 430 Super I/O Board Instruction Manual"

Ohio Scientific Instruments, 11679 Hayden St., Hiram, OH 44234
"Model 420C, 4K Memory Expansion Board"
Instruction Manual - use together with OSI Application Note No. 2 on the 480 Backplane and Application Note No. 5 on interfacing OSI boards to other systems including KIM-1.

ON-LINE, 24695 Santa Cruz Hwy., Los Gatos, CA 95030
This classified ad newsletter often announces KIM-1 and 6502 software and hardware accessories. 18 issued $3.75.

Helmers, Carl, "There's More to Blinking Lights Than Meets the Eye"
Byte 1, No. 5, pp. 52-54 (January 1976)
A program for creating patterns of flashing lights (LEDs).

Lloyd, Robert G., "There's More to Blinking Lights, etc."
KIM-1/6502 Users Notes
A KIM-1 version of Carl Helmers earlier program in Byte.

✓ Ziegler, John, "Breakpoint Routine for 6502"
Dr Dobbs Journal 1, No. 3, pp. 17-19 (1976)
Requires a terminal and a TIM Monitor. Upon entering, the program counter is printed, followed by the active flags, accumulator, register, Y register and stack pointer.

Anon., "What's New Kim-o-sabee?"
Byte 1, No. 8, p. 14 (April 1976)
Brief notes on KIM-1.

Espinosa, Chris, "A String Output Subroutine for the 6502"
DDJ 1, No. 8, p. 33 (September 1976)
This routine saves pointers, loops, etc. in outputting the string.

Meier, Marcel, "6502 String Output, Revisited"
DDJ 1, No. 10, p. 50 (November 1976)
Further mod of Espinosa's earlier routine.

ANON., "Control Logic for Microprocessor Enables Single Step"
Electronic Design, p. 78 (October 11, 1976)
Uses 6502 system.

ANON., "6502 Disassembler"
Interface Age, p. 14 (September 1976)

Butterfield, Jim, "KIM Goes to the Moon"
Byte 2, No. 4, pp. 8-9, 132 (April 1977)
A lunar lander program; see also same program in KIM-1/6502 users notes.

Hybrid Technologies, P.O. Box 163, Burnham, PA 17009
"Ad for KIM-1 Peripherals"
Byte 2, No. 8, p. 157 (August 1977)
2K/8K ROM based, EProm Programmer, 2K/4K/8K Ram boards, assembler board, TV Interface board, relay board, mother boards.

Laabs, John, "Build a $20 EPROM Programmer"
Kilobaud No. 9, pp. 70-77, (Sept 1977)
KIM-1 is used to run software and some external hardware to program the 5204 4K EPROM.

Ohio Scientific Instruments, Hiram, Ohio, 44234, "A Computer that Thinks in BASIC"
Kilobaud No. 9, p. 10, (Sept 1977)
Announcement of OSI's Model 500 CPU board built on 6502. Complete with 8K Basic in ROM for $298.

Clarke, Sheila, "A PET for Every Home"
Kilobaud No. 9, pp. 40-42, (Sept 1977)
A look at the Commodore PET 2001 based on the 6502. About $600 includes Video terminal keyboard, 12K, (8K Basic in ROM and 4K operating system).

American Institute for Professional Education, Carnegie Bldg., Hillcrest Road, Madison, N. J., 07940, "Microprocessing Fundamentals"
Circular Advertisement - approx. Aug 15, 1977.
Dr. Joseph B. Ross, Purdue Univ. and Dr. Garnett Hill, Oklahoma State Univ. will present a course in Fall of 1977 at several locations. Course is based on KIM-1 hardware together with instruction in Digital Devices, Programing Fundamentals, Advanced Programing, Peripherals, 1/0 addressing, applications, etc. Cost about $600 including a KIM-1 to keep after the course.

Gregson, Wilfred J. II, "RTTY with the KIM"
73 Magazine 9 No. 204, p 110-112 (Sept 1977)
A clever program for using KIM-1 and the 6-digit LED display as a readout for a RTTY signal. Simply feed the audio from a receiver into the tape input of KIM-1 and read the message as it flows across the display (about 45.5 baud, 60 wpm). Can also handle other ratio to 100 baud). Can also use KIM-1 as a display only, operating from an already available terminal unit.

Bumgarner, John O., "A-KIM-1 Sidereal/Solar Clock"
Interface Age 2 No. 9, p-36-37 (Aug 1977)

Atkins, R. Travis, "A New Dress for KIM"
Byte 2 No. 9, p-26-27 (Sept 1977)
Describes mounting the KIM-1 in a briefcase together with power supply, prototype boards, etc.

Chamberlin, Hal, "A Sampling of Techniques for Computer Performance of Music"
Byte 2 No. 9, p-62-83 (Sept 1977)
General Discussion of Music Generation plus detailed information on application to KIM-1 and a description of the hardware and software for a D/A music board and software package being marketed by Micro Technology Unlimited, 29 Mead St., Manchester, N.J., 03104. PC board alone is $6.00, assembled and tested D/A board $35.00, software package on KIM cassette is $13.00 additional.

Beals, Gene, PO Box 371, Montgomeryville, PA 18936, "User Group for the Commodore PET 2001 Computer"
Ref: On Line 2 No. 11 pg 2 (Aug 24, 1977)
Yearly membership $5.00 brings Users Notes publication.

Cater, J., 11620 Whisper Trail, San Antonio TX 78230, "Run OSI 6502 8K Basic on your TIM or JOLT"
On Line 2 No. 11, p. 3 (Aug 24, 1977)
Cost $4.00 for annotated source and object code of patches for TIM or JOLT."

Firth, Mike, 104 N. St. Mary, Dallas, Texas 75214, "Large Type Summary of Command Coder for 6502 plus addresses."
On Line 2 No. 11, p. 8 (Aug 24, 2977)
Cost: $0.13 stamp plus SASE.

House, Gil, PO Box 158, Clarksburg, Md., 20734, "6502 Legible Tape Labeler."
On Line 2 No. 11, p. 9 (Aug 24, 1977)
A program for TIM (JOLT DEMON), Hex tape and documentation $4.00

cont. on pg.21

## TTY RAPID LOAD

Markus P.Goenner,Buel,3205 Mauss,Switzerland

```
0000    D8          SIMPLD   CLD
0001    A9 00                LDA #$00
0003    85 F8                STA INL
0005    85 F9                STA INH
0007    20 2F 1E             JSR CRLF       PROGRAM-START: 0000
000A    20 5A 1E    ADDR     JSR GETCH
000D    C9 0D                CMP #'CR       PROGRAM DESCRIPTION:
000F    F0 05                BEQ DATA       AFTER YOU HIT THE "G"-KEY ON THE TTY,THE PROGRAM
0011    20 AC 1F             JSR PACK       ANSWERS WITH A "CR-LF".
0014    F0 F4                BEQ ADDR       ENTER NOW THE ADDRESS WHERE YOU WISH TO LOAD DATA.
0016    A5 F8       DATA     LDA INL        LEADING ZERO'S NEED NOT TO BE ENTERED FOR THE
0018    85 FA                STA POINTL     ADDRESS FIELD.ON A "CR" FROM YOU,THE TTY PROCEED
001A    A5 F9                LDA INH        A "CR-LF" AND YOU ARE READY FOR ENTERING DATA IN
001C    85 FB                STA POINTH     HEXA CODE.JUST ONE BYTE AFTER THE OTHER.AT THE END
001E    20 2F 1E    LINE     JSR CRLF       OF A LINE,TYPE A "CR".TO JUMP BACK IN THE MONITOR,
0021    20 5A 1E    INPUT    JSR GETCH      TYPE AN "ESC" AND THE TERMINAL WILL PRINT A DOLLAR
0024    C9 0D                CMP #'CR       SIGN BEFORE A "CR-LF" AND THEN YOU ARE BACK IN THE
0026    F0 F6                BEQ LINE       KIM-MONITOR.
0028    C9 1B                CMP #'ESC      BY THE WAY,THE PROGRAM IS FULLY RELOCATABLE.
002A    D0 0B                BNE STORE
002C    A9 24                LDA #'$        0000 D8 G
002E    20 A0 1E             JSR OUTCH      0200
0031    20 2F 1E             JSR CRLF       D8A90085F885F9202F1E205A1EC90DF00520AC1FF0F4A5F885
0034    4C 64 1C             JMP CLEAR      FAA5F985FB202F1E205A1EC90DF0F6C91BD00BA92420A01E20
0037    20 AC 1F    STORE    JSR PACK       2F1E4C641C20AC1FD0E5205A1E20AC1FA000A5F891FA20631F
003A    D0 E5                BNE INPUT      1890D3$
003C    20 5A 1E             JSR GETCH
003F    20 AC 1F             JSR PACK
0042    A0 00                LDY #$00
0044    A5 F8                LDA INL
0046    91 FA                STA (POINTL),Y
0048    20 63 1F             JSR INCPT
004B    18                   CLC
004C    90 D3                BCC INPUT
```

*Add this to t[illegible]al Time Clock program from issue #4.*
*If you didn't [illegible] #4, then you'll find the RTC in THE*
*FIRST BOOK OF KIM.......the editor*

Charles H. Parsons
80 Longview Rd.
Monroe Conn  06468

This is the temperature control I mentioned. That's about it for now. All this could be expanded or consolidated if desired.

I thought you might be interested in one thing which gave me a lot of trouble. When comparing the current temperature with the table I first tried to use BMI. This worked most of the time and then at a certain point it fell through. The trouble was that this is meant to be used with signed arithmetic and does not work if the subtraction results in a number that looks like a signed negative number. Switching to BCC cleared this up. Its easy enough to say " Look at the manual" but if you think you are doing the right thing this does not occur to you immediatly. I don't know if others have fallen into this trap but I thought it was worth mentioning.

Read Temperature Once Per Minute

```
Line   Code     Label    Instruction          Comment

0100   A581     TKTEMP   LDA SEC              Do At 50TH Second
0102   29FC              AND FC
0104   C950              CMP #$50
0106   F001              BEQ DO
0108   60                RTS
0109   208001   DO       JSR FREQ             Read Frequency At PB1
010C   A581              LDA SEC
010E   29FC              AND FC               Capture For 4 Seconds
0110   C950              CMP #$50
0112   F0F5              BEQ DO
0114   F8                SED                  Work In Decimal
0115   38                SEC
0116   A5F9              LDA INH              Get LSB's Of Frequency
0118   8596              STA CFREQL           Put In Current Frequency
011A   E594              SBC LCAL             Subtract Calibration
011C   8589              STA CTEMPL           Put In Current Temperature
011E   A5FA              LDA POINTL           Repeat For MSB'S
0120   8597              STA CFREQH
0122   E595              SBC HCAL
0124   858A              STA CTEMPH
0126   B00F              BCS POS              Exit If Result Is Positive
0128   A900              LDA #$00             Complement If Negative
012A   38                SEC
012B   E589              SBC CTEMPL
012D   8589              STA CTEMPL
012F   A900              LDA #$00
0131   E58A              SBC CTEMPH
0133   09C0              ORA #$C0             And Put CX In CTEMPH
0135   858A              STA CTEMPH
0137   D8       POS      CLD                  Go Back To HEX
0138   60                RTS                  Exit
```

Additional Zero Page Locations

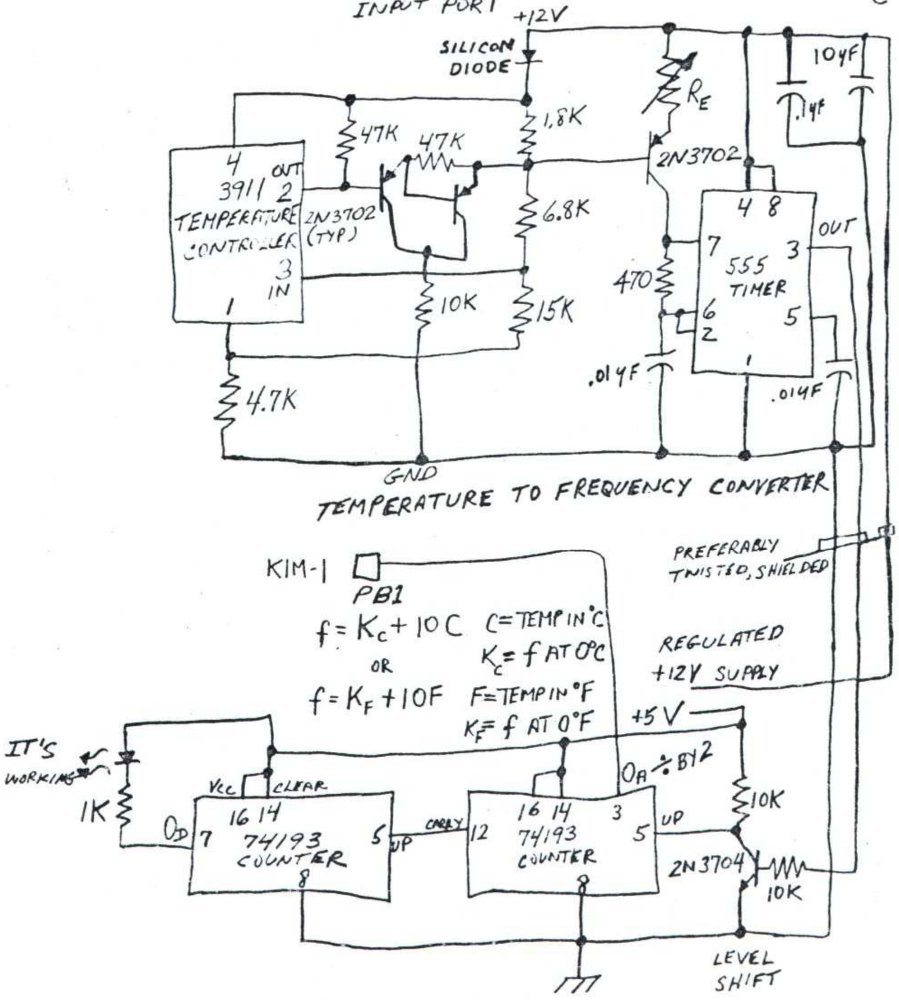
```
0089          CTEMPL                    LSB'S Of Current Temperature
008A          CTEMPH                    MSB'S Of Current Temperature
0094          LCAL                      LSB'S Of Calibration Constant
0095          HCAL                      MSB'S Of Calibration Constant
0096          CFREQL                    LSB'S Of Current Frequency
0097          CFREQH                    MSB'S Of Current Frequency
```

This is a subroutine which when added to the clock display routine will read the input port PB1 exery minute at the 50TH second and subtract the calibration constant in zero page locations The calibration constant is the frequency at zero degree's.

ADJUST VALUE OF $R_E$ FOR $\Delta f/^{\circ}$ AT KIM INPUT PORT

$R_E \approx$ 10K FOR 10 Hz/°F

$R_E \approx$ 18K FOR 10 Hz/°C



TEMPERATURE TO FREQUENCY CONVERTER

$f = K_C + 10C$  C = TEMP IN °C, $K_C = f$ AT 0°C

OR

$f = K_F + 10F$  F = TEMP IN °F, $K_F = f$ AT 0°F

KIM INTERFACE

C.H. PARSONS 3-20-77

## Twentyfour Hour Conversion

```
Line Code   Label  Instruction     Comment
1780 A582   HRA    LDA MIN       Do On The Hour
1782 D017          BNE OUTN
1784 A483          LDY HR        If Hour Is 12
1786 C012          CMP #$12      Set To Zero
1788 D002          BNE N
178A A000          LDY #$00
178C A584   N      LDA DAY       If Afternoon
178E 2901          AND #$01      Add 12
1790 F006          BEQ OK
1792 F8            SED
1793 18            CLC
1794 98            TYA
1795 6912          ADC #$12
1797 A8            TAY           Put In 24 Hour
1798 8498   OK     STY ALTHR     Counter
179A D8            CLD
179B 60     OUTN   RTS
```

### Additional Zero Page Locations

```
0098        ALTHR              24 Hour Counter
```

This is a subroutine which generates a 24 hour clock. This is more convenient for control applications. This program could be incorporated in the clock interrupt routine if it were rewritten.

## Display Current Temperature While 2 On KIM Is Pressed

```
Line   Code     Label    Instruction       Comment

0140   206A1F   DSTEMP   JSR GETKEY      Do When 2 Is Pressed
0143   C902              CMP #$02
0145   D02D              BNE RTS1
0147   A97F              LDA #$7F        Set Output Ports
0149   8D4117            STA PADD
014C   A20D              LDX #$0D        Initial Digit Number
014E   A002              LDY #$02        Output Two Bytes
0150   A589              LDA CTEMPL      Output Absolute Value Of
0152   85F9              STA INH         Temperature
0154   A58A              LDA CTEMPH
0156   293F              AND #$3F        Mask Sign
0158   85FA              STA POINTL
015A   20281F            JSR SCAND1      Display Temperature
015D   A58A              LDA CTEMPH
015F   29C0              AND #$C0        Minus?
0161   F00A              BEQ PLUS
0163   A07F              LDY #$7F        If So Superimpose Minus Sign
0165   8C4117            STY PADD        Set Input Ports
0168   A20B              LDX #$0B
016A   204E1F            JSR CONVD +6
016D   A900     PLUS     LDA #$00        Set Input Ports
016F   8D4117            STA PADD
0172   F0CC              BEQ DSTEMP      Do Again
0174   60       RTS1     RTS
```

This is a subroutine which when added to the clock display routine will display the current temperature on the KIM-1 display while 2 on the KIM-1 keyboard is depressed.

## Temperature Control

```
Line Code     Label  Instruction      Comment

00B0 A581     CNTRLT LDA  SEC         Do On The Minute
00B2 D033            BNE  OUTZ
00B4 A000            LDY  #$00        Get Temperature
00B6 A69A            LDX  TEMP
00B8 A58A            LDA  CTEMPH
00BA 29C0            AND  #$C0        If Minus Set To
00BC F002            BEQ  ARND        Zero
00BE A200            LDX  #$00
00C0 A598     ARND   LDA  ALTHR       Select Day Or Night
00C2 C59F            CMP  DAYST       Table Of Set Points
00C4 9004            BCC  NITE
00C6 C5A0            CMP  DAYEND
00C8 9002            BCC  BGN
00CA A00A     NITE   LDY  #$0A
00CC 8A       BGN    TXA
00CD A200            LDX  #$00
00CF D19B     LP     CMP  (TAB1),Y
00D1 D00B            BCC  OUTP        If Temperature Preceeds
00D3 C8              INY              Set Point,Output
00D4 E8              INX              Proper Control Code
00D5 E00A            CPX  #$0A        If Not Keep Looking
00D7 D0F6            BNE  LP          Through Table To
00D9 A9FF     OUTP   LDA  #$FF        To The End
00DB 8D0117          STA  PADD
00DE 8A              TXA
00DF A8              TAY
00E0 B19D            LDA  (TAB2),Y
00E2 8D0017          STA  PAD         PA-0 Thru PA-7 Are
00E5 85A1            STA  COUT        Output Ports
00E7 60       OUTZ   RTS
```

### Tables

```
17C1          TAB1                    Temperature Set
----                                  Points TD1-TDA
17CA
17CB                                  Temperature Set
----                                  Points TN1-TNA
17D4
17D5          TAB2                    Control Codes
----
17DF
```

## Temperature Control (continued)

### Additional Zero Page Locations

```
Line Code     Label  Instruction      Comment

009B C1                               Temperature Table
009C 17                               Pointers
009D D5                               Control Table
009E 17                               Pointers
009F          DAYST                   Start Of Day Table
00A0          DAYEND                  End Of Day Table
00A1          COUT                    Current Control Code
```

This is a subroutine which puts a word at an output port which is determined by set points in a table. Refer to the work sheet for details.

## Work Sheet For Temperature Control

| | | Alarm | | Heat | | Vent | | Fan | | Code |
|---|---|---|---|---|---|---|---|---|---|---|
| | | on | off | on | off | on | off | on | off | |
| Output Port | | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 | |
| Temperature Range | Boundary<br>Day Nite | | | | | | | | | |
| | <TD1<TN1 | | | | | | | | | |
| 1 | Too Cld | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | A5 |
| | TD1 TN1 | | | | | | | | | |
| 2 | Hyst. | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 25 |
| | TD2 TN2 | | | | | | | | | |
| 3 | Cold | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 65 |
| | TD3 TN3 | | | | | | | | | |
| 4 | Hyst. | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 45 |
| | TD4 TN4 | | | | | | | | | |
| 5 | Normal | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 55 |
| | TD5 TN5 | | | | | | | | | |
| 6 | Hyst: | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 51 |
| | TD6 TN6 | | | | | | | | | |
| 7 | Warm | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 59 |
| | TD7 TN7 | | | | | | | | | |
| 8 | Hyst. | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 58 |
| | TD8 TN8 | | | | | | | | | |
| 9 | Warmer | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 5A |
| | TD9 TN9 | | | | | | | | | |
| 10 | Hyst. | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1A |
| | TDA TNA | | | | | | | | | |
| 11 | Too Hot | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 9A |
| | >TDA>TNA | | | | | | | | | |

This is an example of a simple temperature control using four devices hooked to an eight bit output port. TD1-TDA & TN1-TNA represent the maximum temperatures in each temperature range. They are located in a table.

The lines labeled Hyst. are interposed between lines where action is taken to provide hysteresis between the on and off points of a device. They may not be necessary in a slow system but might be desirable in a fast system with tight control.

The code shown represents the proper word to place at the output port for proper control in any temperature range.

Each pair of outputs would be connected to a flip-flop for control of the respective devices.

### Pack Temperature into 1 Byte Of Hybrid Code

```
Line Code   Label  Instruction        Comment

179C A581   PKTEMP LDA  SEC      Do On The Minute
179E D020          BNE  OUTF
17A0 A589          LDA  CTEMPL   Divide By Ten
17A2 4A            LSR
17A3 4A            LSR
17A4 4A            LSR
17A5 4A            LSR
17A6 859A          STA  TEMP
17A8 A58A          LDA  CTEMPH   Use FF for overflow
17AA C916          CMP  #$16     At 160 Degrees
17AC 9004          BCC  #$04
17AE A9FF          LDA  #$FF
17B0 859A          STA  TEMP
17B2 18            CLC           Multiply CTEMPH
17B3 0A            ASL           By Ten
17B4 0A            ASL
17B5 0A            ASL
17B6 0A            ASL
```

```
17B7 9003            BCC SKIP    Test For Over 100
17B9 18              CLC         If So Convert MSB'S
17BA 69A0            ADC #$A0    To Hexadecimal
17BC 059A    SKIP    OBA TEMP    And Combine ½ Bytes
17BE 859A            STA TEMP
17C0 60      OUTF    RTS
```

Additional Zero Page Locations

```
009A         TEMP                Compressed Temperature
```

Although the temperature given by CTEMP is completely general it requires two bytes to describe. In order to reduce this to one byte and still provide a quasi-understandable code a hybrid notation was chosen. This code is limited to 0-159 degrees. The four LSB'S are retained in decimal notation and the four MSB'S are converted to hexadecimal.

ex. D6=136 degrees

Below 100 the temperatures can be read as decimal.

Frequency Counter Subroutine

```
Line Code   Label  Instruction      Comment

0180 A901   FREQ   LDA #$01     Set I/O Ports
0182 8D0317        STA PBDD
0185 A581          LDA SEC      Do For 4 Seconds
0187 A8            TAY
0188 2903          AND #$03
018A F038          BEQ BACK
018C 98            TYA
018D 2902          AND #$02     Display For Seconds
018F D030          BNE DSPL     3&4
0191 A900          LDA #$00     Zero Frequency Counter
0193 85F9          STA INH      And Count For Second 2
0195 85FA          STA POINTL
0197 85FB          STA POINTH
0199 F8            SED
019A AD0217 L      LDA PBD      Stall For One Pulse
019D 2902          AND #$02
019F D0F9          BNE L
01A1 AD0217 H      LDA PBD
01A4 2902          AND #$02
01A6 F0F9          BEQ H
01A8 18            CLC          Count One Pulse
01A9 A901          LDA #$01
01AB 65F9          ADC INH
01AD 85F9          STA INH
01AF A900          LDA #$00
01B1 65FA          ADC POINTL
01B3 85FA          STA PIONTL
01B5 A900          LDA #$00
01B7 65FB          ADC PIONTH
01B9 85FB          STA POINTH
01BB A581          LDA SEC      Still Second 2?
01BD 2901          AND #$01
01BF D0D9          BNE L        If So Keep Counting
01C1 201F1F DSPL   JSR SCANDS   Display Count
01C4 60     BACK   RTS
01C5 200003 RFREQ  JSR KIM      Start Here To Update
01C8 208001        JSR FREQ     Every 4 Seconds
01CB 18            CLC
01CC 90F7          BCC RFREQ    Loop
```

This is a subroutine which can be run by itself by entering at 01C5 or under program control with JSR FREQ. The output is the frequency at PB1 in Hertz.

end

# A KIM BINARY DUMP + LOAD ROUTINE

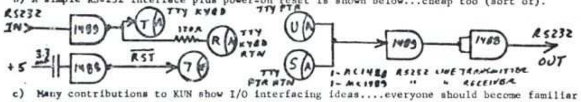
FROM..

John Oliver
Associate Professor
of Astronomy
Williamson Hall
University of Florida
Gainesville FL 32611

Well, I guess the time has come to stop enjoying the good stuff others have sent in and to start contributing myself. The enclosed program was written for SPICA (Small Portable Interactive Computer for Astronomy) to allow dumping and loading blocks of data (or code) under program control. I have put in lots of comments and it should be almost self explainatory. The user defines a buffer area and dumps or loads that area at a rate of about 1000 bytes in 12 seconds. If an incoming file exceeds the buffer length reading stops when the buffer is filled and an error flag is set. If the incoming file ID does not match the requested ID the buffer is filled and an error flag is set. We have a relay on one output line connected to the REMOTE jack on the recorder to start and stop the tape. (Soon we hope to use a PHIDEC recorder for better control.) I use as much of the KIM ROM as possible but I wish they had used more subroutines in there, its not as nice as it could have been. With these subroutines a $29 cassette recorder can become a useful digital data recorder at resonably high data rates (100 bytes per second + housekeeping).

Other Misc. Comments: a) We have used SUPERTAPE and SUPERDUMP/LOAD on a Radio Shack CTR-29 and a Radio Shack Minisette-V (very nice because of the CUE feature) with few problems. With the Minisette-V we need to unplug the earphone when recording to get sucess- I have not good reason why ???? But others might watch out.

b) A simple RS-232 interface plus power-on reset is shown below...cheap too (sort of).



c) Many contributions to KUN show I/O interfacing ideas....everyone should become familiar with the Motorola 68XX line of support chips (get their good data book). A major virtue of the 6502 is that it is compatible with all that good Motorola stuff....ignore M's instructions to gate the addressing with VMA since address is always valid with the 6502. I have used the 6820 (PIA: 16 I/O lines plus 4 handshaking control lines) and the 6850 (ACIA: good for interface to a terminal or a large computer terminal port. They are coming out with floppy disk and tape recorder support chips soon.....I couldn't wait and am using a NEC floppy controller meant for an 8080 (ugh) but wish I had waited.

d) My 9 year old, Jennifer Anne Oliver, loves WUMPUS and thanks you for publishing it..She runs KIM like a pro, they sure learn young.

```
; ************* SUPERDUMP/SUPERLOAD BY JOHN P. OLIVER ************
;                DEPARTMENT OF PHYSICS AND ASTRONOMY
;               UNIVERSITY OF FLORIDA .GAINESVILLE FL
;
;    THIS PROGRAM ALLOWS THE USE OF THE KIM-1 CASSETTE TAPE
;  INTERFACE TO READ AND WRITE DATA BLOCKS UNDER PROGRAM CONTROL.
;  IT IS DERIVED FROM JIM BUTTERFIELD'S SUPERTAPE ROUTINES IN
;  KIM USERS NOTES #2 BUT EACH DATA BYTE IS WRITTEN AS AN 8-BIT
;  CHARACTER RATHER THAN AS TWO ASCII CODED HEX CHARACTERS.  THUS
;  1K BYTES ARE DUMPED OR LOADED IN LESS THAN 12 SECONDS.  THE
;  TAPE FORMAT HAS BEEN SOMEWHAT CHANGED IN THAT THE NUMEBER
;  OF BYTES IN THE RECORD ARE WRITTEN IN PLACE OF SAL/H. KIM ROM
;  ROUTINES ARE USED AS FAR AS POSSIBLE WHILE KEEPING FULL
;  SUBROUTINE STATUS FOR THESE PROGRAMS.
;
;  TO WRITE A FILE:  PUT STARTING ADDRESS IN $17F5/6
;                    PUT ENDING ADDRESS + 1 IN $17F7/8
;                    PUT FILE ID IN $17F9
;
;  THEN JSR SUPERD.  THIS ROUTINE CAN BE INTERRUPTED AS LONG
;  AS THE INTERRUPT ROUTINES DO NOT TOTAL MORE THAN 100
;  MICROSECONDS IN EACH 200 MICROSECONDS.
;
;  TO READ A FILE:   PUT INPUT BUFFER ADDRESS IN $17F5/6
;                    PUT END OF BUFFER + 1 IN $17F7/8
;                    PUT DESIRED FILE ID IN $17F9 (USE $00 TO GET
;                    NEXT FILE, REGARDLESS OF ITS ID ON TAPE)
;
;  THEN JSR SUPERL.  THE PROGRAM WILL RETURN WITH THE DATA IN
;  THE BUFFER AREA.THE RECEIVED ID IN $17F9, AND A FLAG ($00CB):
;                      = 00             LOAD OK
;                      = FF OR = 7F     BUFFER OVERRUN
;                      = FE OR = 7E     CHECKSUM ERROR
;
;  A FILE ID ERROR YIELDS 80, 7F, OR 7E.
;
;  THE LOAD ROUTINE IS RELOCATABLE. TO RELOCATE THE DUMP ROUTINE
;  MODIFY THE JSR'S TO  OUTCHT, OUTCH, OUTBT, AND HEXTA.
;
;  ANY TAPE RECORDER CONTROL ROUTINES CAN BE CALLED BEFORE SUPERL
;  OR SUPERD.
;
;  NOTE: SUPERL WILL NOT RETURN TO THE CALLING ROUTINE IF THE TAPE
;  IS NOT READING PROPERLY.
```

```
00C8                     ORG  $00C8
00C8   00       DESID    FCB  0        ;INTENDED INPUT ID
00C9   00       EALB     FCB  0        ;BUFFER END ADDRESS
00CA   00       EAHB     FCB  0
00CB   00       LFLG     FCB  0        ;LOAD FLAG WORD
00CC   00       GANG     FCB  0
00CD   00       TIC      FCB  0
00CE   00       COUNT    FCB  0
00CF   00       TRIB     FCB  0
00D0   02       NPUL     FCB  02
00D1   C3       TIMG     FCB  $C3
00D2   03                FCB  $03
00D3   7E                FCB  $7E
0100                     ORG  $0100    ;SUPERDUMP STARTS AT $0100
0100   A9 AD    SUPERD   LDA  #$AD     ;'STA' OP CODE FOR VEB
0102   8D EC17           STA  VEB
0105   20 3219           JSR  INTVEB   ;INITIALIZE VEB
0108   A9 27             LDA  #$27
010A   85 CC             STA  GANG     ;SBD OUTPUT WORD
010C   A9 BF             LDA  #$BF     ;OPEN CHANNELS
010E   8D 4317           STA  PBDD
0111   A9 20             LDA  #$20     ;SEND 32 SYNC CHARACTERS
0113   85 CD             STA  TIC      ;SAVE CHAR COUNT
0115   A9 16             LDA  #$16     ;....SYNC ....
0117   48       HIC1     PHA           ;SAVE CHARACTER
0118   20 9001           JSR  OUTCHT
011B   68                PLA           ;RESTORE CHARACTER
011C   C6 CD             DEC  TIC      ;REDUCE COUNTER
011E   D0 F7             BNE  HIC1     ;FINISHED?
0120   A9 2A             LDA  #$2A     ;SEND '*'
0122   20 9001           JSR  OUTCHT
0125   A9 00             LDA  #$00
0127   20 6E01           JSR  OUTBT
012A   38                SEC           ;COMPUTE # OF BYTES ....
012B   AD F717           LDA  EAL      ;.... TO BE SENT ....
012E   ED F517           SBC  SAL      ;....SAVE NBL ....
0131   48                PHA           ;....TEMP ON STACK
0132   AD F817           LDA  EAH
0135   ED F617           SBC  SAH
0138   20 6E01           JSR  OUTBT    ;OUTPUT NBH
013B   68                PLA           ;GET NBL AND ....
013C   20 6E01           JSR  OUTBT    ;....OUTPUT
013F   AD F917           LDA  ID       ;SEND....
0142   20 6E01           JSR  OUTBT    ;....ID
0145   20 EC17  SUPDP1   JSR  VEB      ;GET BYTE USING VEB ....
0148   20 8D01           JSR  OUTCHC   ;....AND SEND IT
014B   20 EA19           JSR  INCVEB   ;INCREMENT FOR NEXT BYTE
014E   AD ED17           LDA  VEB+1    ;ARE WE AT ....
0151   CD F717           CMP  EAL      ;.... END ADDRESS?
0154   AD EE17           LDA  VEB+2
0157   ED F817           SBC  EAH
015A   90 E9             BCC  SUPDP1   ;NOT FINISHED,GET MORE
015C   A9 2F             LDA  #$2F     ;SEND '/'
015E   20 9001           JSR  OUTCHT
0161   AD E717           LDA  CHKL
0164   20 6E01           JSR  OUTBT    ;SEND CHECKSUM
0167   AD E817           LDA  CHKH
016A   20 6E01           JSR  OUTBT
016D   60                RTS
016E   48       OUTBT    PHA                ;HEX OUTPUT ROUTINE;SAVE BYTE
016F   4A                LSR
0170   4A                LSR
0171   4A                LSR
0172   4A                LSR
0173   20 8101           JSR  HEXTA    ;GET 4 MSB AS ASCII
0176   20 9001           JSR  OUTCHT   ;WRITE IT
0179   68                PLA           ;RESTORE BYTE
017A   20 8101           JSR  HEXTA    ;GET 4 LSB AS ASCII
017D   20 9001           JSR  OUTCHT   ;WRITE IT
0180   60                RTS
0181   29 0F    HEXTA    AND  #$0F     ;MASK OFF 4 LSB
0183   C9 0A             CMP  #$0A
0185   18                CLC
0186   30 02             BMI  HEXTA1
0188   69 07             ADC  #$07     ;A TO F
018A   69 30    HEXTA1   ADC  #$30     ;0 TO 9
018C   60                RTS
018D   20 4C19  OUTCHC   JSR  CHKT     ;CHECKSUM CALCULATION
0190   A0 08    OUTCHT   LDY  #$08     ;SET FOR 8BITS
0192   84 CE             STY  COUNT    ;SAVEBIT COUNT
0194   A0 02    TRY      LDY  #$02     ;SET FOR 3PHASES
0196   84 CF             STY  TRIB     ;SAVEPHASE COUNT
0198   B6 D0    ZON      LDX  NPUL,Y   ;# OF 1/2 CYCLES
019A   48                PHA           ;SAVE CHARACTER
019B   78       ZON1     SEI           ;DISABLE INTERRUPTS
019C   2C 4717  ZON2     BIT  CLKRDI   ;TIMER DONE?
019F   10 FB             BPL  ZON2     ;NO,WAIT
01A1   B9 D100           LDA  TIMG,Y   ;GET WAIT TIME IN MICRO ....
01A4   8D 4417           STA  CLKIT    ;....SECONDS FOR TIMER
01A7   A5 CC             LDA  GANG     ;FLIP OUTPUTBIT ....
01A9   49 80             EOR  #$80     ;.... BETWEEN 0AND 1
01AB   8D 4217           STA  SBD      ;OUTPUTBIT
01AE   58                CLI           ;ENABLE INTERRUPTS
01AF   85 CC             STA  GANG     ;SAVE OUTPUTBIT
01B1   CA                DEX           ;ALL CYCLES SENT?
01B2   D0 E7             BNE  ZON1     ;NO,SEND MORE
01B4   68                PLA           ;RESTORE CHARACTER
```

```
0185  C6 CF                DEC  TRIB    ;ONE LESSPHASE TO GO
0187  F0 C5                BEQ  SETZ    ;AND THIS ISPHASE 3
0189  30 07                BMI  ROUT    ;ALLPHASES DONE

018B  4A                   LSR          ;GETBIT ....
018C  90 DA                BCC  ZON     ;.... IF IT IS '1' ....
018E  A0 C0       SETZ     LDY  #$C0    ;.... CHANGE TO 2400 HZ
01C0  F0 D6                BEQ  ZON     ;FORCED BRANCH
01C2  C6 CE       ROUT     DEC  COUNT   ;ONE LESSBIT TO GO
01C4  D0 CE                BNE  TRY
01C6  60                   RTS          ;ALL DONE
0200                       ORG  $0200   ;SUPERLOAD STARTS AT $0200
0200  AD F917     SUPERL   LDA  ID      ;STORE....
0203  85 C8                STA  DESID   ;....INTENDED ID
0205  AD F717              LDA  EAL     ;STORE BUFFER END ADDRESS
0208  85 C9                STA  EALB
020A  AD F817              LDA  EAH
020D  85 CA                STA  EAHB
020F  A9 00                LDA  #$00    ;INITIALIZE ....
0211  85 CB                STA  LFLG    ;.... LOAD ERROR FLAG
0213  8D F917              STA  ID      ;....AND ID FIELD
0216  A9 60                LDA  #$60    ;'RTS' OPCODE
0218  8D EC17              STA  VEB     ;RETURN OUT OF LOADT
021B  20 8C18              JSR  $188C   ;PUSH PATCH ADDRESS ONSTACK. GO TO LOADT
021E  8D F917     PATCH    STA  ID      ;WE GET HERE FROM $190C...JMP VEB
0221  C5 CB                CMP  DESID   ;INTENDED ID ?
0223  F0 0A                BEQ  PATCH2  ;YES
0225  A9 00                LDA  #$00    ;ANY ID ....
0227  C5 CB                CMP  DESID   ;.... OK?
0229  F0 04                BEQ  PATCH2  ;YES
022B  A9 80                LDA  #$80    ;SET ERROR FLAG ....
022D  85 CB                STA  LFLG    ; .... AND CONTINUE
022F  A9 8D       PATCH2   LDA  #$8D    ;'STA' OPCODE
0231  8D EC17              STA  VEB     ;RECREATE VEB STORE INST
0234  18                   CLC          ;CLEAR CARRY FOR ENDING ADD COMP
0235  AD EE17              LDA  VEB+2   ;GET # OF BYTES _ ....
0238  7D F517              ADC  SAL     ;.... ADD SAL ....
023B  8D F717              STA  EAL     ;.... TO GET EAL
023E  AD ED17              LDA  VEB+1   ;GET # OF BYTES 4 ....
0241  7D F617              ADC  SAH     ;.... ADD SAH ....
0244  8D F817              STA  EAH     ;.... TO GET EAH
0247  20 3219              JSR  INTVEB  ;CLEAR CHKSUM.SET UP VEB
024A  20 241A     PATCH1   JSR  RDCHT   ;GET NEXT BYTE (ACC HAS 7BIT ASCII) ....
024D  AD EA17              LDA  SAVX+1  ;.... SO GET THE FULL 8BIT BYTE
0250  20 4C19              JSR  CHKT    ;ADD TO CHECK SUM
0253  20 EC17              JSR  VEB     ;STORE IT
0256  20 EA19              JSR  INCVEB  ;INCREMENT VEB ADDRESS FOR STORE
0259  AD ED17              LDA  VEB+1   ;END ADDRESS?
025C  C5 C9                CMP  EALB    ;BUFFER END ?
025E  D0 02                BNE  PATCH3  ;NO
0260  F0 05                BEQ  PATCH4  ;MAYBE ?
0262  CD F717     PATCH3   CMP  EAL     ;RECORD END?
0265  D0 E3                BNE  PATCH1  ;NO. GET MORE BYTES
0267  AD EE17     PATCH4   LDA  VEB+2
026A  C5 CA                CMP  EAHB    ;BUFFER END ?
026C  D0 0F                BNE  PATCH5  ;NO
026E  CD F817              CMP  EAH     ;ALSO RECORD END ?
0271  D0 28                BNE  ERROR2  ;NO. ERROR EXIT
0273  AD ED17              LDA  VEB+1   ;LOW ORDER BYTE ALSO OK?
0276  CD F717              CMP  EAL
0279  D0 20                BNE  ERROR2  ;NO; ERROR EXIT
027B  F0 05                BEQ  PATCH6
027D  CD F817     PATCH5   CMP  EAH     ;RECORD END ?
0280  D0 C8                BNE PATCH1   ;NO. CONTINUE
0282  20 241A     PATCH6   JSR  RDCHT   ;GET ENDING CHARACTER
0285  C9 2F                CMP  #$2F    ;'/'?
0287  D0 10                BNE  ERROR
0289  20 F319              JSR  RDBYT   ;GET CHECKSUM LO
028C  CD E717              CMP  CHKL    ;CHECKSUM OK?
028F  D0 C8                BNE  ERROR
0291  20 F319              JSR  RDBYT   ;GET CHECKSUM HI
0294  CD E817              CMP  CHKH    ;
0297  F0 04                BEQ  EXIT
0299  C6 CB       ERROR    DEC  LFLG    ;FE OR 7E = CHECKSUM ERROR
029B  C6 CB       ERROR2   DEC  LFLG    ;FF OR 7F = OVERRUN ERROR
                  ; 80. 7E OR 7F INDICATES ID ERROR
                           RTS          ;RETURN
```

## KIMSI COMMENTS

From the response I've received concerning the KIM to S-100 bus adapter being offered by FORETHOUGHT PRODUCTS, I'd say there are a number of satisfied users. Nothing but words of praise for the product, so far. With S-100 memory running as low as $125 for 8K kits (BASE 2), the scheme seems like a reasonable method for system expansion. As far as assembled S-100 boards are concerned, the only ones that I am familiar with are the KENT-MOORE products. They market video and memory boards which seem to work as well as they look.

more

By the way, I've been informed that FORETHOUGH PRODUCTS have cleared up any problems with their telephone service and are now accepting VISA (BankAmericard). Their phone number is (503) 485-8575. They indicate off-the-shelf delivery.

BASE 2 INC, PO Box 9941, Marina del Ray, Ca 90291   (213) 822-4499

KENT-MOORE INSTRUMENT CO., PO Box 507, Industrial Ave, Pioneer, Oh 43554
(419) 737-2352

FORETHOUGHT PRODUCTS, PO Box 386, Coburg, Or., 97401

************************

## RANDOM ACCESS CORNER

Here's a new feature of the NOTES for those who have special needs...

PEN PAL NEEDED - P. A. Ras, H. Gorterhof 138, DELFT, NETHERLANDS
Mr. Ras also needs info on Friden Flexowriter/KIM interfacing.

BURROUGHS TERMINAL/KIM-1 INTERFACE info needed by Gene Moore, 817 Windsor Rd
Cumberland, Md.21502

BRINGING UP 8K OSI BASIC ON KIM? or trying to bring it up?...get in touch with Donald Hill, 60 Evans Ave., East Hartford, Ct. 06118

FORTRAN II FOR THE 6502---"We're thinking about offering it depending on interest. Send SASE and info on what software you need to GENESEE MICROCOMPUTERS, 29 Genesee St., Piffard NY 14533"

GERMAN USER GROUP GETTING STARTED in the Frankfurt area. For more info, contact Erich Scheiber, Berliner St. 10, 6236 Eschborn, West Germany.

KIM-3 and/or KIM-4 desperately needed!!! contact JOHNSON COMPUTER
(216) 725-4560

WASHINGTON AREA KIM ENTHUSIASTS who are interested in starting a KIM KLUB, send a S.A.S.E. or call!!! WAKE c/o Ted Beach, 5112 Williamsburg Blvd, Arlington, Va 22207   (703) 538-2303

MICRO-SOFTWARE SPECIALISTS INC., 1911 Meadow Lane, Arlington, Tx 76010 have announced that they have cleared up the problems with their assembler mentioned in our newsletter. They are accepting VISA at (817) 274-0291

WANTED: KIM-2 or KIM-3 RAM board for memory expansion. Contact Kenneth W. Ensele, 1337 Foster Rd., Napa Ca 94558  (707) 226-5014

FOR SALE: KIM-1 and expermintation accessories used in TERC microprocessor workshops. Valued at $500.00, will sell for $300.00.
W. L. Sadler, 2020 Easy Street, Waukesha, Wi., 53186
(414) 547-9391

************************

BOOK REVIEW SECTION from Charles A. Mills, 677 Lippincott Ave.,
Moorestown, N.J. 08057

UNIQUE PROGRAMMING BOOK *** HOW TO PROGRAM MICROCOMPUTERS by William Barden (SAMS $8.95) explains looping, stacks, list processing, bit manipulation, etc. The unique feature is that all program explanations are for the 8080, 6800, and 6502 so one can see how each is programmed to do the same thing. Twenty utility programs in each system are provided for comparison of coding requirements.

(Ive seen this book and can also recommend it.....ERIC)

continued from pg. 15

✓ Simpson, Richard S., "A Date with KIM"
Byte 1, No. 9, pp. 8-12 (May 1976)
Description of the features of KIM-1.

Microcomputer Associates, 111 Main St., Los Altos, CA 94022
"Jolt Microcomputer"
Radio-Electronics 47, No. 6, p. 66 (June 1976)
Includes description of JOLT, based on 6502, and gives demonstration program using DEMON Monitor.

Travis, T. E., "KIM-1 Microcomputer Module"
Microtrek, pp. 7-16 (August 1976)
Notes and programs for KIM-1 including Drunk test and several useful routines.

Anon., "MOS Technology - KIM MCS 6502"
Interface Age 1, No. 9, pp. 12, 14 (August 1976)
An announcement of the KIM-1.

✓ Rankin, Roy and Wozniak, Steve, "Floating Point Routines for the 6502"
Dr Dobbs Journal 1, No. 7, pp. 17-19 (August 1976)
Calculations from $10^{-38}$ to $10^{+38}$ with 7 significant digits.

✓ Bradshaw, Jack, "Monitor for the 6502"
Dr Dobbs Journal 1, No. 7, pp. 20-21 (August 1976)
Monitor a la OSI.

✓ Garetz, Mark, "Lunar Lander for the 6502"
Dr Dobbs Journal 1, No. 7, pp. 22-25 (August 1976)
A game requiring TIM Monitor and a terminal.

Gupta, Yogesh M., "True Confessions: How I Relate to KIM"
Byte 1, No. 12, pp. 44-48 (August 1976)
A series of notes on KIM-1. Includes Clock Stretch and Random Access Memories, Bus Expansion and modification of drive capability using tristate drivers, Interrupt Prioritizing Logic and Halt Instruction.

Thompson, Geo. L., "KIM on, Now"
Byte 1, No. 13, pp. 93-94 (September 1976)
Notes on using KIM-1.

✓ Wozniak, Steve, "Mastermind: A Number Game for the 6502"
DDJ 1, No. 8, pp. 26-27 (September 1976)
A number game adaptable to KIM-1 with terminal.

Baum, Allen and Wozniak, Stephen, "A 6502 Dissembler"
Interface Age 1, No. 10, pp. 14-23 (September 1976)

Kjeldsen, Tony, "Next of KIM" (letter)
Byte 1, No. 14, p. 136 (October 1976)

✓ Pittman, Tom, "Tiny Basic for 6502"
DDJ 1, No. 9, pp. 22-23 (October 1976)
Available from Itty Bitty Computers. TB650K (0200-OAFF) is for KIM and most homebrew 6502 systems with RAM in first 4K of memory.

Anon., "Build a Simple A to D"
Interface Age 1, No. 12, pp. 12-14 (November 1976)
Simple circuit, 6502 software, 16 locations. Use to interface a pot or a joystick.

Pollock, James W., "1000 WPM Morse Code Typer"
73 Mag. No. 196, pp. 100-103 (January 1977)
Use of KIM-1 for sending code at 9-1000 WPM from a keyboard.

Robbins, Carl M., "The Microprocessor and Repeater Control"
QST 61, No. 1, pp. 30-34 (January 1977)
KIM-1 control of repeater functions.

Cushman, Robert H., "Bare-bones Development Systems Make Good Learning Tools"
EDN 22, No. 6 (March 20, 1977)
See also 22, No. 8, pp. 104-111 (April 20, 1977)
22, No. 4, pp. 89-92 (February 20, 1977)
22, No. 10, pp. 84-90 (May 20, 1977)
22, No. 12, pp. 79-84 (June 20, 1977)
Use of KIM-1 in a music program is detailed in April 1977 issue.

Salter, Richard J. and Burham, Ralph W., "Navigation with Mini-O"
Byte 2, No. 4, pp. 100-109 (April 1977); See also Byte 2, No. 2, p. 62 (February 1977) and Byte 2, No. 3, p. 70 (March 1977).
Several articles in a series on the Omega Navigation System and the Mini-O Receiver driven by a KIM-1 processor. Developed at the Ohio University Avionics Engineering Center.

✓ Haas, Bob, "KIM-1 Memory Expansion"
Kilobaud, No. 4, pp. 74-76 (April 1977)
Adding the S.D. Sales 4K Low Power RAM board to KIM-1.

Gordon, H. T., "Stringout Mods"
DDJ 2, No. 2, p. 8 (February 1977)
A 6502 program applicable to KIM-1 to relocate blocks of instructions in RAMs.

Sherman, Ralph, "A 650X Program Relocater"
DDJ 2, No. 4, pp. 30-31 (April 1977)

Ockers, Stan, "TV Sketch Program"
DDJ 2, No. 4, pp. 32-33 (April 1977)
A program for use with KIM-1 equipped with a Southwest Tech Prod Co. Graphics Board GT 6144.

Simpson, Rick, "Come Fly with KIM"
✓ Byte 2, No. 6, pp. 76-80 (June 1977)
Load 12K of memory in two minutes with a "Fly Reader" for paper tape.

✓ Lancaster, Don, "A TVT for your KIM"
Kilobaud, No. 6, pp. 50-63 (June 1977)
TVT-6L is a low cost method of providing a TV monitor for KIM-1. Uses minimum new hardware but depends on a software program in KIM-1 memory for handling characters. Uses a low cost TV (Pansonic T-126A) for monitor.

NO Lancaster, Don, "Build the TVT-6"
Popular Electronics 12, No. 1, pp. 47-52
A low cost direct video display based on KIM-1 software and a minimum of added hardware. Slightly different than the TVT-6L.

Pickles and Trout, P.O. Box 2270, Goleta, CA 93018 "TV Mod Kit"
Detailed instructions and kit of parts for conversion of a low cost ($80 approx.) Hitachi SX Chassis (Model P-04, P-08, PA-8, etc.) for a TV Monitor.

Grater, Robert, "Giving KIM Some Fancy Jewels"
Byte 2, No. 7, pp. 126-127 (July 1977)
Adding a remote LED display for the KIM-1.

✓ Runyan, Grant, "The Great TV to CRT Monitor Conversion"
Kilobaud, No. 7, pp. 30-31 (July 1977)
Although not specific to KIM-1, this article is useful in adapting a monitor to KIM. Uses inexpensive 12" Hitachi Model P-04, P-08, PA-4, PA-8. See also Sams Photofact Folder 1 Set 1601 or Folder 3 Set 1501.

✓ Fish, Larry, "Troubleshoot Your Software"
Kilobaud, No. 8, pp. 112-113 (August 1977)
A trace program for 6502.

more next time...

JOHNSON
COMPUTER (216) 725-4560

## PRELIMINARY INFORMATION ON MICROSOFT 8K BASIC FOR KIM-1

Variable names must start with an alphabetic character,eg. A, A1, A(3,7,2), ZULU
String (literal) variable. names are followed by a dollar sign,eg. A$, ZULU$, A$(2,3)
Although variable names may consist of more than two characters, only the first two characters uniquely identify the varable,eg. COST is the same as CORE

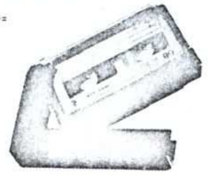
OPERATORS: -, +, *, /, ↑, *NOT*, *AND*, *OR*, >, <, <>, <=, =, >=

STATEMENTS
CLEAR
DATA
DEF
DIM
END
FOR
GOTO
GOSUB
IF...GOTO
IF...THEN
INPUT
LET
NEXT
ON...GOTO
ON...GOSUB
POKE
PRINT or ?
READ
REM
RESTORE
RETURN
STOP

FUNCTIONS
ABS(X)
ATN(X)
COS(X)
EXP(X)
FRE(X)
INT(X)
LOG(X)
PEEK(X)
POS(I)
RND(X)
SGN(X)
SIN(X)
SPC(I)
SQR(X)
TAB(I)
TAN(X)
USR(I)

STRING FUNCTIONS
ASC(X$)
CHR$(I)
FRE(X$)
LEFT$(X$)
LEN(X$)
MID$(X$,I,J)
RIGHT$(X$,I)
STR$(X)
VAL(X$)

COMMANDS
CONT
LIST
NEW
NULL
RUN



@ Erase typed line
SHIFT/O or ← Erase last character
: Seperates statements on same line
CONTROL/C Interrupts execution or listing
CONTROL/O Inhibits output to terminal

Both versions of BASIC use page zero and page one. They start at 2000HEX. Although they are meant to be used with serial terminals, I/O pointer locations are provided. The USER, PEEK, POKE, and WAIT statements are used to link BASIC to machine code programs and the KIM-1 ports. The 6 digit version uses two-letter symbols for error messages. The nine digit version spells out complete error messages. When executions or listings are interrupted by means of the CONTROL/C or an error, BASIC indicates the number of the line it was about to execute or list.

| CAT # | PRECISION | LOADS AT | # OF BYTES | MIN. SYSTEM RAM | RANGE | PRICE |
|---|---|---|---|---|---|---|
| KB-6 | 6 DIGITS | 2000HEX | 8257 | 12000 | 10E-32 to 10E+32 | 97.50* |
| KB-9 | 9 DIGITS | 2000HEX | 8802 | 12000 | 10E-32 to 10E+32 | 129.00* |

*TERMS: PAYMENT WITH ORDER. ADD $4.00 FOR SHIPPING AND HANDLING. OHIO RESIDENTS ADD 4.5% SALES TAX ( $4.39 for KB-6 and $5.81 for KB-9 )

Microsoft 8K BASIC for the KIM-1 is furnished on cassette with complete documentation, including a 239 page Schaum's Outline Series' Theory and Problems of Programming with BASIC by Byron S. Gottfried, Ph.D., McGraw Hill.

P. O. BOX 523 MEDINA, OHIO 44258

Kim-1/6502 USER NOTES
% ERIC C. REHNKE
109 CENTRE AVE
W. NORRITON, PA. 19401.
U.S.A.

DELPHIA, PA
M
NOV 19
1977

MIDNIGHT RIDE • ONE IF BY LAND, TWO IF BY SEA
USA 24c

#7 8

FIRST CLASS

JAMES STORK
3867 A MIRAMAR ST
LA JOLLA, CA 92037