

KIM-1/6502 USER NOTES

MARCH 1977

SEE YOU IN DAYTON !!!

ISSUE 4

YOU WILL BE THERE- WON'T YOU ???

Time really flies when you're having fun! (or are really busy) It's hard to believe that four issues of the NOTES have been published already. I can still remember when the first subscriptions started rolling in and now there are over 800 KIM aficionados in the group with no signs of tapering off.

The format of our little journal is in a state of flux-as you can see. The booklet form seemed like a good idea until I got feedback from a number of you indicating that something a little easier to punch and insert in a binder would be a little more convenient. Well, here it is. I hope this will improve things.

Don Lancaster's really been busy with his KIM-1! Two national hobbyist magazines will be featuring Lancaster's KIM TV typewriter circuits this summer. Watch Kilobaud magazine in issue #6 or #7 and check out Popular Electronics for July and August.

Rumor has it that Mr. Lancaster is also working on a KIM graphics interface. His latest book (bible?), CMOS COOKBOOK, will be reviewed in an upcoming issue of the USER NOTES.

Robert Cushman, Special Features Editor for EDN (one of the top industrial electronics magazines), has started a series of tutorial articles on microsystem design procedures that look to be very informative. Cushman, also a member of our KIM-1 User Group, wants people to start thinking in terms of system design rather than just function design and will evidently be using KIM in design examples.

More and more computer clubs have KIM-1 special interest groups. Here's two more:

Long Island Computer Association (LICA) contact KIM-1 Coordinator-Steve Perry, 6 Brookhaven Drive, Rocky Point, N.Y. 11778 516-744-6462 after 7 pm.

Amateur Computer Group of New Jersey-contact 650X group coordinator-John Loofbourrow at 233-7068 (area code unknown).

PUT THIS ON YOUR SOCIAL CALENDAR**

The second annual COMPUTERFEST '77 (June 10, 11, 12 - Cleveland Ohio) will be held at the Bond Court Hotel, 777 St. Clair Avenue in downtown Cleveland. An Admission charge of \$2.00 will be good for a weekend of manufacturers exhibits, seminars, tech sessions, a flea market etc. For more information- send a S.A.S.E. to Midwest Affiliation of Computer Clubs, P.O. Box 83, Brecksville, Ohio 44141.

(M.A.C.C.)

CORRECTION TO ISSUE #3-Cass Lewart informed me that on page 7, the mnemonic in address location 22 should be LDY #yy (not LX #yy), the machine code (AO) is correct.

KIM-1 USER NOTES is published every 5 to 8 weeks. The subscription rate for U.S. and Canadian subscribers is \$5.00 for issues 1 thru 6 including 1st class postage. Foreign subscriptions - \$8.00 including 1st class air mail postage.

Payment should be made in U.S. funds with a check or money order (no cash or purchase orders) please.

KIM-1 USER NOTES
c/o Eric C. Rehnke
425 Meadow Lane
Seven Hills, Ohio 44131 (Phone - 216-524-7241)

To alleviate possible typographical errors, please try to submit articles in original type, single spaced on white bond so that we may cut and paste instead of retyping. Also, if you expect a personal response to correspondence, please include a self addressed stamped envelope, to help defray expenses.

Note on locations OOF1 and OOF2..When you hit GO, the contents of OOF1 transfer to the status register, and P2 to the stack pointer. Always preset OOF1 to 00 to avoid being accidentally in decimal mode; and OOF2 to FF to avoid having the stack "write over" your page 1 programs or data.
.. Jim Butterfield

KIM-1 TO S-100 BUS ADAPTER

PAGE 1

Got a flyer from Forethought Products. They announced KIMSI, an 8-slot motherboard that would enable most S-100 type boards to be used with KIM. They say that all decoding and buffering circuitry is provided. Get more info from Forethought Products, P.O. Box 386, Coburg, Ore., 97401.

KIM-1 SOFTWARE PACKAGES

Robert Tripp, author of the PLEASE package, mentioned that he is making four more KIM-1 software packages available soon. Tripp says his packages, known as HELP, will include a text editor, a mailing list handler, a form letter writing aid, and an information retrieval system. For more info, write The Computerist, P.O. Box 3, Chelmsford, Mass. 01824. ask for HELP.

TO NEW SUBSCRIBERS

At least one of you, who recently subscribed to our Notes, did not get all three back issues. They came apart in route and the Post Office sent back the pieces. We are now using envelopes for mailing back issues 'cuz we want to be sure no one misses any data. Please contact me if you were shorted one or two back issues recently.....

CW RECEIVE ROUTINE

22 magazine, April '77 (page 80) has a morse code interpreter program that may be of interest to you hams. It was written for the 6800 but could be adapted to KIM with little work.

To convert your receiver's audio output to a digital signal so your computer can work on it, you need some type of filtering and digitizing circuitry. A circuit of this type was included in an article which appeared in Popular Electronics, January '77 (page 37). The complete circuit for the signal conditioner could consist of IC 1, 2, 3, and 5 from the schematic on page 39.

If any of you are working along these lines, let's hear from you.

MORE ON THE SERIAL A'DAPTOR BOARD SAB-1

Bob Grater had an article in Kilobaud magazine issue #1 (page 114) which explained the SAB-1 with a full schematic and interface details. If you're adapting a parallel input TVT to your machine and want it to look like a terminal, check this out.

KIM-1 ACCESSORIES MARKET

I've had conversations with several manufacturers who will be marketing accessories for KIM shortly. Among these items will be an optical bar code scanner and software loader, several enclosures, boards for the KIM-4 etc. As soon as formal product announcements are received, they will be passed along in the Notes. I will not evaluate these products or even infer that they actually exist until I've seen them.

It sounds like KIM is really taking hold in the marketplace.

LET ME KNOW YOUR OPINION OF THIS TYPE NEWSLETTER FORMAT!

HEY RTTY'rs - THERE IS AN AUTO-START NET ON 80 METERS
(3637.5 KHZ \pm 10 HZ) THAT INCLUDES SOME KIM-1's. FOR MORE
INFO, CONTACT TRUMAN BOERKOEL KBJUG,
2050 BROOKRIDGE DR., DAYTON, OHIO 45431

Ever long for an assembler? Remember when you wrote that 300 byte program - and discovered that you'd forgotten one vital instruction in the middle? And to make room, you'd have to change all those branches, all those addresses... Or the program with that neat piece of coding in it, that you suddenly need to remove (say, to change it to a subroutine) ... but if you do, you'll have to fill all that empty space with NOPs? It's enough to make a grown programmer cry...

Dry those tears. Program RELOCATE will fix up all those addresses and branches for you, whether you're opening out a program to fit in an extra instruction, closing up space you don't need, or just moving the whole thing someplace else.

RELOCATE doesn't move the data. It just fixes up the addresses before you make the move. It won't touch zero page addresses; you'll want them to stay the same. And be careful: it won't warn you if a branch instruction goes out of range.

You'll have to give RELOCATE a lot of information about your program:

- (1) Where your program starts. This is the first instruction in your whole program (including the part that doesn't move). RELOCATE has to look through your whole program, instruction by instruction, correcting addresses and branches where necessary. Be sure your program is a continuous series of instructions (don't mix data in; RELOCATE will take a data value of 10 as a BPL instruction and try to correct the branch address), and place a dud instruction (FF) behind your last program instruction. This tells RELOCATE where to stop.

Place the program start address in locations EA and EB, low order first as usual. Don't forget the FF behind the last instruction; it doesn't matter if you temporarily wipe out a byte of data - you can always put it back later.

- (2) Where relocation starts. This is the first address in your program that you want to move. If you're moving the whole program, it will be the same as the program start address, above. This address is called the boundary.

Place the boundary address in locations EC and ED, low order first.

- (3) How far you will want to relocate information above the boundary. This value is called the increment. For example, if you want to open up three more locations in your program, the increment will be 0003. If you want to close up four addresses, the increment will be FFFC (effectively, a negative number).

Place the increment value in locations EE and E9, low order first.

- (4) A page limit, above which relocation should be disabled. For example, if you're working on a program in the 0200 to 03FF range, your program might also address a timer or I/C registers, and might call subroutines in the monitor. You don't want these addresses relocated, even though they are above the boundary! So your page limit would be 17, since these addresses are all over 1700.

On the other hand, if you have memory expansion and your program is at address 2000 and up, your page limit will need to be much higher. You'd normally set the page limit to FF, the highest page in memory.

Place the page limit in location E7.

Now you're ready to go. Set RELOCATE's start address, hit go - and ZAP! - your addresses are fixed up.

After the run, it's a good idea to check the address now in 00EA and 00EB - it should point at the FF at the end of your program, confirming that the run went OK.

Now you can move the program. If you have lots of memory to spare, you can write a general MOVE program and link it in to RELOCATE, so as to do the whole job in one shot.

But if, like me, you're memory-deprived, you'll likely want to run RELOCATE first, and then load in a little custom-written program to do the actual moving. The program will vary depending on which way you want to move, how far, and how much memory is to be moved. In a pinch, you can use the FF option of the cassette input program to move your program.

Last note: the program terminates with a BRK instruction. Be sure your interrupt vector (at 17FE and 17FF) is set to KIM address 1C00 so that you get a valid 'halt'.

6502 Program: RELOCATE
February, 1977

Jim Butterfield
14 Brooklyn Avenue
Toronto, Ontario M4M 2K5

```

; following addresses must be initialized
; by user prior to run
00E7 PAGLIM **+1 limit above which kill relocn
00E8 ADJUST **+2 adjustment distance (signed)
00EA POINT **+2 start of program
00EC BOUND **+2 lower boundary for adjustment
; main program starts here
0110 D8 START CLD
0111 A0 00 LDY #0
0113 B1 EA LDA (POINT),Y get op code
0115 A8 TAY + cache in Y
0116 A2 07 LDX #7
0118 98 LOOP TYA restore op code
0119 3D 8E 01 AND TAB1-1,X remove unwanted bits
011C 5D 95 01 EOR TAB2-1,X & test the rest
011F F0 03 BEQ FOUND
0121 CA DEX on to the next test
0122 D0 F4 BNE LOOP ... if any
0124 EC 9D 01 FOUND LDY TAB3,X length or flag
0127 30 0D BMI TRIP triple length?
0129 F0 22 BEQ BRAN branch?
012B E6 EA SKIP INC POINT moving right along..
012D D0 02 BNE INEX ..to next op code
012F E6 EB INC POINT+1
0131 88 INEX DEY
0132 D0 F7 BNE SKIP
0134 F0 DA BEQ START
; length 3 or illegal
0136 C8 TRIP INY
0137 30 D9 BMI START+2 illegal/end to BRK halt
0139 C8 INY set Y to 1
013A B1 EA LDA (POINT),Y lo-order operand
013C AA TAX ... into X reg
013D C8 INY Y=2

```


PAGE 2

I REALLY FLIPPED WHEN I SAW THIS PROGRAM — BUTTERFIELD DOES IT AGAIN!! ECR

```

013E B1 EA      LDA (POINT),Y      hi-order operand
0140 20 79 01   JSR ADJUST        change address, maybe
0143 91 FA      STA (POINT),Y      ...and put it back
0145 88         DEY                Y=1
0146 8A         TXA
0147 91 FA      STA (POINT),Y      ...also hi-order
0149 A0 03      LDY #3            Y=3
014B 10 DE      BPL SKIP
; branch: check 'to' and 'from' addresses
014D C8         BRAN              Y=1
014E A6 EA      LDX POINT        'from' addr lo-order
0150 A5 EB      LDA POINT+1      .. & hi-order
0152 20 79 01   JSR ADJUST        change, maybe
0155 86 E0      STX ALOC         save lo-order only
0157 A2 FF      LDX #FFF         flag for 'back' branches
0159 B1 EA      LDA (POINT),Y    get relative branch
015B 18         CLC
015C 69 02      ADC #2           adjust the offset
015E 30 01      BMI OVER        backwards branch?
0160 E8         INX              nope
0161 86 E3      OVER STX LIMIT
0163 18         CLC
0164 65 EA      ADC POINT        calculate 'to' lo-order
0166 AA         TAX              .. and put in X
0167 A5 E3      LDA LIMIT        00 or FF
0169 65 EB      ADC POINT+1      'to' hi-order
016B 20 79 01   JSR ADJUST        change, maybe
016E CA         DEX              readjust the offset
016F CA         DEX
0170 8A         TXA
0171 38         SEC
0172 E5 E0      SEC ALOC         recalculate relative branch
0174 91 EA      STA (POINT),Y    and re-insert
0176 C8         INY              Y=2
0177 10 B2      BPL SKIP
; examine address and adjust, maybe
0179 C5 F7      ADJUST CMP PAGLIM
017B B0 11      BCS OUT          too high?
017D C5 ED      CMP BOUND+1
017F D0 02      BNE TES2         high-order?
0181 E4 EC      CPX BOUND        lo-order?
0183 90 09      TES2 BCC OUT      too low?
0185 48         PHA              stack hi-order
0186 8A         TXA
0187 18         CLC
0188 65 E8      ADC ADJUST        adjust lo-order
018A AA         TAX
018B 68         PLA              unstack hi-order
018C 65 E9      ADC ADJUST+1      and adjust
018E 60         OUT              RTS
; tables for op-code identification
018F 0C 1F 0D   TAB1 .BYTE $0C,$1F,$0D,$87,$1F,$FF,$03
0192 87 1F FF
0195 03
0196 0C 19 08   TAB2 .BYTE $0C,$19,$08,$00,$10,$20,$03
0199 00 10 20
019C 03
019D 02 FF FF   TAB3 .BYTE 02,$FF,$FF,$01,$01,$00,$FF,$FE
01A0 01 01 00
01A3 FF FE
; end

```

Credit for the concept of RELOCATE goes to Stan Ockers, who insisted that it was badly needed, and maintained despite my misgivings that it should be quite straightforward to program. He was right on both counts.

THANKS TO JIM + STAN !!

MOVIN' - How to move data or programs around Jim Butterfield

Here's a few little programs/procedures to use when you want to move memory contents around. They fit in anywhere.

In the next two programs XX XX means the 'from' address minus one; TT TT means the 'to' address minus one. In both cases, these are the starting addresses of your data. NN is the total number of locations to be moved. Check the examples if this isn't clear.

(1) Move 1-256 bytes to a higher address:

A2 NN HD XX XX 9D TT TT CA D0 F7 00

Example: move contents of 0234-0278 to 0258-028D

A2 45 HD 33 02 9D 57 02 CA D0 F7 00

(2) Move 1-256 bytes to a lower address:

A2 00 E8 HD XX XX 9D TT TT E0 NN D0 F5 00

Example: move contents of 0258-0288 to 0234-0274

A2 00 E8 HD 57 02 9D 33 02 E0 31 D0 F5 00

(3) Move over 256 bytes:

I recommend writing the data you want to move onto a fresh cassette tape.

Now, put the address where you want the data into locations 1775-6 (low order first, as always). Put FF into location 1779 and perform a tape read.

Tom Wear

Dear Eric:

Per your query for info on 74LS145, I purchased mine from

Active Electronic Sales Corp
P. O. box 1035
Framingham, MA 01701
(617) 879-0077

They stock a most complete list of 74LS chips as well as many other hard-to-find items, like the latest off the production line at Texas Instruments in TTL as well as linears, all grade one--no surplus, rejects and junk. Minimum order \$10.00 plus \$1.00 postage and handling.

Their initial response has been good--7 to 10 days--however on a few occasions "temporary-out-of-stock back-orders" have been neglected. Direct communication with Manager Alan Barroll has solved these oversights quickly.

I exchanged the KIM-1 U4 74145 for the 'LS' version and have adapted an OSI mother board to provide 74LS367 3-State Hex Bus Drivers for the address lines and 8833 for the data lines. I will share this and other hardware and software items as soon as I can produce the legible drawings and write-up (documentation is always the toughest part).

KIM-1 UTILITY: DIRECTORYJim Butterfield
Toronto

Ever thought about the best way to organize your programs on tape? I used to call the first program on each tape number 01, the next 02, etc. Mostly I was afraid of forgetting the ID number and having trouble reading it in. Program DIRECTORY (below) fixes up that part of the problem and liberates you to choose a better numbering scheme.

You've got 254 program IDs to choose from ... enough for most program libraries with some to spare. So why not a little structuring to help you remember what a program is for?

I suggest the following: First digit - 0 to 9 for completed (or 'permanent') programs ... A to F for programs you're still working on; Second digit - 0 to 9 for programs, A to F for data files. Using this scheme, I'd know that ID 5E is a permanent data file; A3 is a program still being writ.

So every program and data file would carry a unique number ... and if you've forgotten what's on a given tape, just run DIRECTORY and get all the IDs.

Another thing that's handy to know is the starting address (SA) of a program, especially if you want to copy it to another tape. (Ending addresses are easy ... just load the program, then look at the contents of 17ED and 17EE). Well, DIRECTORY shows starting addresses, too.

I got the idea for DIRECTORY from Peter Jennings, Toronto, who has a teletype-oriented program to do the same thing. This version uses keyboard/display.

The program is fully relocatable, so put it anywhere convenient. Start at the first instruction (0000 in the listing). Incidentally, 0001 to 001D of this program are functionally identical to the KIM monitor 199C to 19C1.

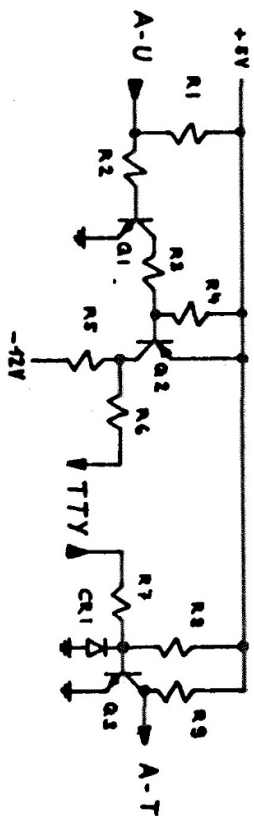
After you start the program, start your audio tape input. When DIRECTORY finds a program, it will display the Start Address (first four digits) and the Program ID. Hit any key and it will scan for the next program.

```

0000 D8      GO      CLD
0001 A9 07      LDA #$07  Directional reg
0003 8D 42 17    STA SBD
0006 20 41 1A SYN JSR RDBIT  Scan thru bits...
0009 46 F9      LSR INH   ..shifting new bit
000B 05 F9      ORA INH   ..into left of
000D 85 F9      STA INH   ..byte INH
000F C9 16      TST      CMP #$16  SYNC character?
0011 D0 F3      BNE SYN   no, back to bits
0013 20 24 1A    JSR RDCHT  get a character
0016 C6 F9      DEC INH   count 22 SYNC's
0018 10 F5      BPL TST
001A C9 2A      CMP #$2A   then test astk
001C D0 F1      BNE TST   ..or SYNC
001E A2 FD      LDX #$FD   if asterisk,
0020 20 F3 19 RD JSR RDBYT  stack 3 bytes
0023 95 FC      STA POINTH+1,X into display
0025 E8         INX       area
0026 30 F8      BMI RD
0028 20 1F 1F SHOW JSR SCANDS ...and shine
002B D0 D3      BNE GO    until keyed
002D F0 F9      BEQ SHOW  at's all folks

```

HERMES AM RS-232 INTERFACE FROM: Martin P. Goemmer
 Dettlbergstr. 107
 8045 Zurich
 Switzerland



Parts list:

Q 1	2 N 2222
Q 2	2 N 2907
Q 3	2 N 2222
CR 1	1 N 914 (1 N 4148)
R 1	1'000 ohms
R 2	5'100 ohms
R 3	2'700 ohms
R 4	1'000 ohms
R 5	560 ohms
R 6	4'700 ohms
R 7	4'700 ohms
R 8	10'000 ohms
R 9	180 ohms

KIMMAZE

by Stan Ockers & Jim Butterfield

Find your way out of the maze. You're the flashing light in the centre of the display. As you move up (key 9), down (key 1), left (4) or right (6), YIM will keep you in the central display; you'll see the walls of the maze moving by as you travel. Like walking through a real maze, you'll only see a small part of the maze as you pass through it. If you can get out, you'll find yourself in a large open area; that means you've won.

Program starts at address 0200.

```

0200 D0      START   CLD
0201 A2 02      LDX #2          3 values
0203 BD B5 02 SETUP LDA INIT,X   from init
0206 95 D2      STA NZPT,X      ..to maze ptr
0203 CA        DEX
0209 10 F0      BIL SETUP
          ;-- pick out specific part of maze
020B A0 0B      MAF   LDY #11
020D B1 D2      GETMOR LDA (NZPT),Y  6 rows x 2
020F 99 D0 00   STA WORK,Y
0212 88        DEY
0213 10 F0      BIL GETMOR
          ;--shift to position vertically
0215 A2 0A      LDX #10        for each of 6 rows..
0217 A4 D4      NXDIG LDY POSIT   shift Y positions
0219 A9 FF      LDA #FF        filling with 'walls'
021B 30        REROL SEC         ..on both sides
021C 36 D9      ROL WORK+1,X
021E 36 D0      ROL WORK,X      roll 'em!
0220 2A        ROL A
0221 88        DEY
0222 D0 F7      BNE REROL
          ;--calculate segments
0224 29 07      AND #7         take 3 bits
0226 A0        TAY            & change to
0227 B9 A0 02   LDA TAB1,Y      segment pattern
022A 95 D0      STA WORK,X      ..and store
022C CA        DEX
022D CA        DEX
022E 10 E7      BIL NXDIG
          ;--test flasher
0230 C6 D5      LIGHT DEC FLUG   time out?
0232 10 0A      BIL MUG         ..no
0234 A9 05      LDA #5         ..yes, reset
0236 85 D5      STA FLUG
0237 A5 DE      LDA WORK+6      ..and..
023A 49 40      EOR #$40       ..flip..
023C 85 DE      STA WORK+6     ...flasher..

          ;--light display
023E A9 7F      MUG   LDA #$7F   open the gate
0240 3D 41 17   STA SADD
0243 A0 09      LDY #309
0245 A2 0A      LDX #10
0247 B5 D9      SHOW  LDA WORK,X  tiptoe thru..
0249 3D 40 17   STA SAD         ..the segments
024C 3C 42 17   STY SBD
024F C6 D6      ST1  DEC STALL   ...pausing
0251 D0 FC      BRE ST1
0253 C3        INY
0254 C3        INY
0255 CA        DEX
0256 CA        DEX
0257 10 EE      BIL SHOW

```

```

      ;--test new key depression
0259 20 40 1F      JSR KEYIN      set dir reg
025C 20 6A 1F      JSR GETKEY     key?..
025F C5 D7         CMP SOK       ..same as last?
0261 F0 CD         BEQ LIGHT
0263 85 D7         STA SOK       no, record it

      ;--test which key
0265 A2 04         LDX #4        5 items in table
0267 DD A8 02 SCAN  CMP TAB2,X
026A F0 05         BEQ FOUND
026C CA           DEX
026D 10 F8         BFL SCAN
026F 30 BC         BMI LIGHT
0271 CA           FOUND DEX
0272 30 8C         BMI START     go key?

      ;--test if wall
0274 BC AD 02      LDY TAB3,X
0277 B9 D8 00      LDA WORK,Y
027A 3D B1 02      AND TAB4,X
027D D0 B1         BRE LIGHT

      ;--move
027F CA           DEX
0280 10 04         BIL NOTUP
0282 C6 D4         DEC POSIT     upward move
0284 D0 85         MLINK BNE MAP  1-o-n-g branch
0286 D0 04         NOTUP BNE SIDEWY
0288 E6 D4         INC POSIT     downward move
028A D0 F8         BNE MLINK
028C CA           SIDEWY DEX
028D D0 06         BNE LEFT
028F C6 D2         DEC MZPT     right move
0291 C6 D2         DEC MZPT
0293 D0 EF         BNE MLINK
0295 E6 D2         LEFT INC MZPT left move
0297 E6 D2         INC MZPT
0299 D0 E9         BNE MLINK

      ;--tables (hex listed)
TAB1 02A0 00 08 40 48 01 09 41 49
TAB2 02A3 13 09 01 06 04
TAB3 02AD 06 06 04 08
TAB4 02B1 01 08 40 40

      ;--sample maze follows
      ;--first 3 bytes are initial cursor poitr
INIT 02B5 B4 02 08
MAZE 02B7 FF FF 04 08 F5 7E 15 00 41 FE 5F 04
      51 7D 5D 04 51 B6 54 14 F7 D5 04 54
      7F 5E 01 00 FD FF 00 00 00 00 00 00
      00 00 00 00 00 00

```

Maze construction: every two bytes, starting at MAZE, represents a complete cross section of the maze; a 1 bit in any position represents a wall.

In the example above, the first cross section is FF FF (all one bits) - this would be an impassable section of wall. The next cross section (04 08) has only two pieces of wall in it, at positions 6 and 13. The zeros at the end represent the 'open space'

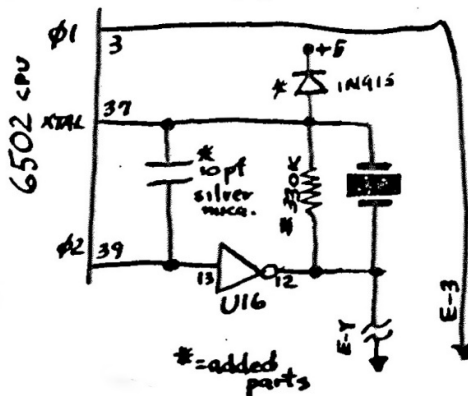
FOR SALE MOS TECHNOLOGY, INC KIM-1 w/1K Homebrew 21102 RAM, POWER SUPPLY 3ea +5V at 1.5A, 1ea -5V at .5A, 1ea +12V at .5A, 1ea -12V at .5A SERIAL ADAPTOR BOARD (SAD-1) (Checked out on a CT 1024 works ok) WOOD CASE EXTRA 22 PIN CONNECTORS w/2 extra on case, CROSS ASSEMBLER MANUAL, TIM MANUEL, PLEASE MONITOR w/TAPE & BOOKS, TINY BASIC w/TAPE & HEX LIST, 10ea MEMOREX 30 Minutes ea side tapes w/PROGRAMS. \$330.00 taken all
 ROBERT G. LLOYD, 7554 SOUTHGATE RD., FAYETTEVILLE, N.C. 28304, (919) 867-5822

This is what J. B. calls "PSEUDATAPE". I did not make it faster because the filter between the PLL and the comparator is designed for the nominal KIM-speed (400 baud) and it will degrade the comparator input if the speed exceeds say 1200 baud. The comparator input waveform becomes triangular and the peak-to-peak value decreases.

I also have a telephone-dialler program (12 digits) that uses KIH's keyboard and display, but since legal problems may arise, I don't find it advisable to publish it. But if anyone are interested, drop me a line.

WUPUS is great, but wouldn't it be fairer to the besst to change 03FF to 02 (and 00A4 to 60)

U4 has been changed from a 74145 to a 74LS145
All 6502 CPUs now have the ROTATE RIGHT (ROR) instruction
The clock circuit has been changed as follows:



A CALCULATOR INTERFACE reworked by the editor

Hooking up a calculator chip to a computer sounded like a neat idea even before I had a computer! For over a year, I have been searching through the available literature for all pertinent information on the subject. Needless to say, my file hasn't exactly overflowed with material. For such a seemingly desirable interface, not much has really been done.

Calculator chip information was hard to get and finding the chips themselves proved even more of a difficulty. It didn't seem worthwhile to use a four function chip as the scientific arrays offered bunches more calculating power for the same amount of work involved.

Recently, the MOS Technology 7529-103 scientific calculator array became available in single quantities. This seemed to be the route to take. The next problem? How do you hook the beast up to Kim?

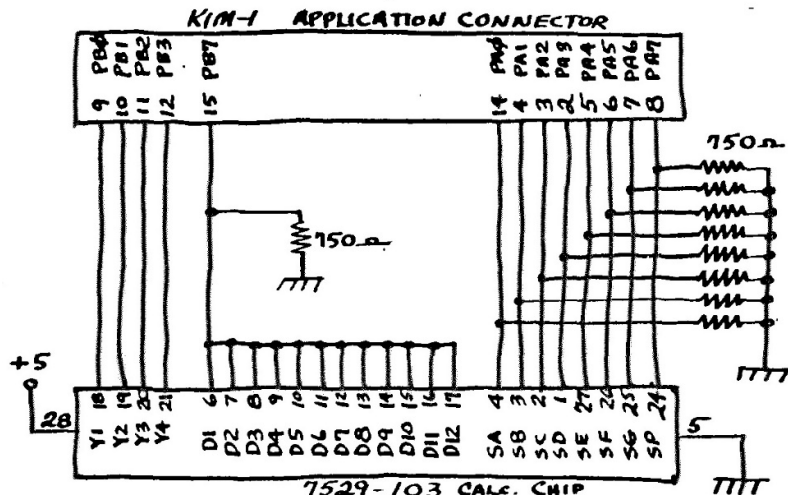
One example of the circuitry necessary to interface the 7529-103 to a micro was presented in Byte (Sept, Oct 1976). This circuit used about 29 IC's to get a two way conversation going with the calculator chip. That's more IC's than there are on Kim! There has to be a better way.

Well, there is a better way to do it. It's called the software approach (replace as much hardware as you can with software). The interface hardware and software driver presented here were originally released as an application note by MOS Technology. One hardware bug and several software bugs were corrected and the thing was modified to work with Kim.

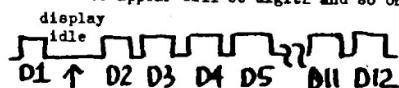
There is one hardware "trick" that you should be aware of: originally the 7529-103 was designed to work with a negative 7.5v supply. If you saw the Byte article, you can see how the chips operating point can be shifted up to use a positive 7.5 volt supply (just reverse the ground and Vdd connections). Now, to make the thing TTL compatible, just lower the positive voltage to +5 volts. This is outside the recommended operating parameters specified by MOS (-6v to -9.5v) but most chips will work alright. (I tried 3 chips and they all operated correctly at +5v). If you bought your chip from Johnson Computer and it doesn't work at +5v - they have assured me that they will exchange your chip for another one.

The device driver starts at 0200, takes a series of specially encoded keystroke data starting from 0300 and handles the input multiplexing and output demultiplexing from the calculator chip. There is a limit of 256 keystrokes and the keystroke data MUST be terminated by \$FF. The answers will be in seven-segment format starting at 0000. This very basic driver does not detect calculator underflow, overflow, or convert the seven-segment data to BCD. It's intended just to get the interface operational-you should be able to improve and/or change it once you understand how it works. Underflow and overflow detection and the BCD conversion routine will be presented in an upcoming issue.

Individual chips may differ slightly in their operating characteristics so the 100 usec. wait loop located at 022C may have to be adjusted. (#14 worked for all the chips I tried). This corresponds to about one-half of a digit strobe.



Since the calculator synchronizes all its I/O functions using the digit strobes, so must the computer... The digit strobes are tied together to give the computer the sync pulses so that it will know the proper time to enter data into the calculator and retrieve the calc. output when done. The computer senses the DISPLAY IDLE time and knows that the next digit strobe to appear will be digit2 and so on....



INPUT KEYCODE DATA

KEY	CODE	KEY	CODE
0	C1	COS	74
1	11	TAN	24
2	21	LN	34
3	31	LOG	44
4	41	\sqrt{x}	54
5	51	RCL	64
6	61	M+	74
7	71	XY	84
8	81	DGR	94
9	91	STO	A4
ARC	A1	CA/CE	B4
.	C2	1/X	C8
-	22	x^2	18
x	32		
10^x	42	10^x	28
e^x	52	e^x	38
=	62	NI	48
(72		
)	82		
π	92	Display	
CHS	A2	Restore	B1
EEX	B2		
SIN	C4		

7-SEGMENT OUTPUT DATA

BARS	HEX
0	3F
0.	BF
1	06
1.	86
2	5B
2.	DB
3	4F
3.	CF
4	66
4.	E6
5	6D
5.	ED
6	7C
6.	FC
7	07
7.	87
8	7F
8.	FF
9	67
9.	E7
BLANK	00
-	40
OF	31
UF	71
.	C0
.	80

by the way - the 7529-103 now costs \$10.00 from Johnson Computer (see last issue) how 'bout that?

CALCULATOR DRIVER

0000 +12 Data Output File
 000C +12 Temp Data Buffer
 0018 Keycode (Temp)
 0019 X Store (Temp)
 1700 PA Data
 1702 PB Data
 1703 PB Control Reg.

```

0200 A9 0F      inits   LDA #0F  initialize PB
0202 8D 03 17    STA 1703
0205 A2 00      exec   LDX #00
0207 BD 00 03    ex 1   LDA,X    read key code
020A 86 19      STX X store
020C 20 23 02    JSR calc
020F A6 19      LDX X store
0211 A5 18      LDA keycode
0213 C9 FF      CMP #ff  check for end
0215 D0 03      BNE more
0217 4C 8C 02    JMP rerang
021A E8         more   INX
021B BD 03      BEQ noff
021D 4C 07 02    JMP ex 1
0220 20 05 1C    noff   JMP out of line
0223 85 18      calc   STA keycode
0225 A0 04      LDY 04  loop count
0227 2C 02 17    A1     BIT PBD  low synch?
022A 30 FB      BMI A1  branch on high
022C A2 14      LDX     wait 100 usec
022E CA         A2     DEX
022F D0 FD      BNE A2
0231 2C 02 17    BIT PBD  low synch?
0234 30 F1      BMI A1
0236 A5 18      LDA     recall keycode
0238 C9 FF      CMP     test for read code
023A F0 34      BEQ read
023C 4A         LSR     right justify
023D 4A         LSR     high order bits
023E 4A         LSR
023F 4A         LSR
0240 AA         TAX     -d- line number
0241 CA         A3     DEX
0242 F0 0C      BEQ write
0244 2C 02 17    A4     BIT PBD  high synch?
0247 10 FB      PBL A4
0249 2C 02 17    A5     BIT PBD  low synch?
024C 30 FB      BMI A5
024E 10 F1      BPL A3
0250 A5 18      write  LDA keycode
0252 29 0F      AND #0F  lower four bits
0254 AA         TAX     only
0255 2C 02 17    B1     BIT PBD  high synch?
0258 10 FB      BPL B1
025A 8E 02 17    STX PBD  write to y-line
025D A2 00      LDX #00
025F 2C 02 17    B2     BIT PBD  low synch?
0262 30 FB      BMI B2
0264 8E 02 17    STX PBD  clear y-line
0267 88         DEY     decr loop count
0268 D0 BD      BNE A1
026A 20 A0 02    B3     JSR delay
026D EA         NOP
026E EA         NOP
026F 60         RTS     ok, then return
  
```

continued on next page

PAGE 5

CALC DRIVER (contd)

```

0270 A0 0B read LDY #0B digits-1
0272 A2 14 C0 LDX 14 wait 100 usec
0274 2C 02 17 C1 BIT PBD high synch?
0277 10 FB BPL C1 not yet?
0279 CA C2 DEX
027A D0 FD BNE C2
027C AD 00 17 LDA PAD read calc. output
027F 99 0C 00 STA store code
0282 88 DEY
0283 30 EA BMI B4
0285 2C 02 17 C3 BIT PBD low synch?
0288 30 FB BMI C3
028A 10 E6 BPL C0
028C A0 01 rearrang LDY 01 rearrange
028E A2 0A LDX 0A digits
0290 B5 0D move LDA 0000,X to
0292 99 00 00 STA 0000,Y proper
0295 C8 INY order...
0296 CA DEX
0297 10 F7 BPL move
0299 A5 0C LDA
029B 85 00 STA
029D 4C 4F 1C JMP back to KIM
02A0 A9 2C delay LDA #2C set up time delay
02A2 8D 05 17 STA CLK8T +8
02A5 2C 07 17 wait BIT CLKKT time out?
02A8 10 FB BPL wait
02AA 2C 00 17 B3B BIT ADAT look for high
02AD 10 FB BPL B3B segment P
02AF 60 RTS back to calc

```

**Thanks to CHRISTOPHER FLYNN FOR HIS HELP
IN DEBUGGING THE DRIVER SOFTWARE !!**

Verify Cassette Tape

James Van Ornum
55 Cornell Drive
Hazlet, NJ 07730

Do you want to verify the cassette tape you just recorded before the information is lost? Then follow this simple procedure:

1. Manually verify that the starting address (\$17F5, \$17F6), the ending address (\$17F7, \$17F8) and the block identification (\$17F9) locations are correct in memory.
2. Enter zeros (\$00) into CHKL (\$17E7) and CHKH (\$17E8).
3. Enter the following routine:

```

17EC CD 00 00 VLB cmp START
17EF D0 03 bne failed
17F4 4C 0F 19 jmp LOAD12
17F4 4C 29 19 failed jmp LOADT9

```

4. Rewind the tape, enter address \$188C, press GO and playback the tape. If the tape compares, the LEDs will come back on with address \$0000. If there is a discrepancy between memory and the tape, the LEDs will come on with address \$FFFF.

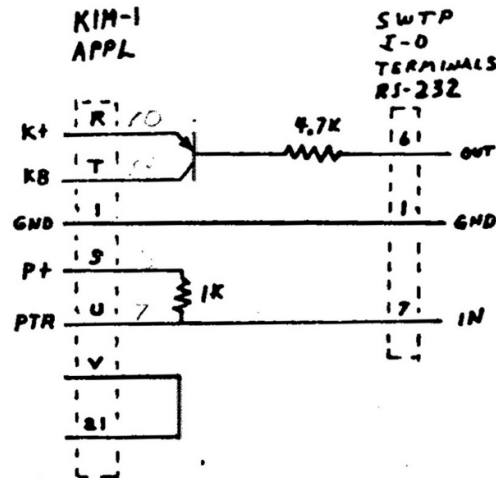
I thoroughly enjoyed HUNT THE WUMPUS in the November 1976 User Notes. However, assembly language source listings are necessary for us to experiment with the programs. I am willing to convert handwritten source listings into typed and assembled versions for inclusion in the User Notes.

PAGE 6

Interface for the SouthWest Technical Products TV typewriter II and KIM-1. The SWTP serial interface board is used. Jumper between terminals V and 21 of KIM must be used. After pressing RESET on KIM type in letter A to start system. Most keyboards do not have a DELETE key. The transistor is a small signal PNP Radio Shack ARCHER package #276-530 (yellow dot). Any small signal PNP should work.

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

WANTED: ANY DATA ON CONSTRUCTING
A SIMPLE TVT FROM SCRATCH—
DAN GARNER
11825 BEACH BLVD
STANTON, CAL
90680



Cass R. Levart
12 Georgian Dr.
Holmdel, N.J. 07733

Variable Speed and Light Control

The basic AC Triac interface described in the January issue of the KIM-1 Newsletter (p.8) can also be used with a slight modification for light dimming, motor speed control, heater settings etc by means of Pulse Width Modulation technique. Using the circuit shown here and the following program one can vary the on/off time ratio of the Triac. Depending which key is depressed determines the width of the ON pulse within a fixed time interval and the average conductivity of the Triac. The program could easily be modified for example to slowly dim a light during a slide show or to accelerate a model train.

Note: we found lights to flicker at certain brightness settings. Please let me know if somebody comes with an improved circuit and/or program.

00 D8	CLD	17 10 FB	BPL ①
01 A9 15	② LDA #15 <i>start 6.7ms</i>	19 A5 2E	LDA 2E
03 85 2E	STA 2E	1B F0 E4	BEQ ② <i>check if 6.7ms are up</i>
05 A9 FF	LDA #FF	1D C6 2E	DEC 2E <i>decrement counter</i>
07 8D 01 17	STA 1701 <i>set PAO to ON</i>	1F 20 6A 1F	JSR GETKEY <i>check which key</i>
0A A9 01	LDA #01	22 C5 2E	CMP 2E <i>counter not equal</i>
0C 8D 00 17	STA 1700 <i>set PAO to OFF</i>	24 D0 E9	BNE ③
0F A9 05	③ LDA #05 <i>start and set time to 0.5ms</i>	26 A9 00	LDA #00 <i>set PAO (triac) to OFF</i>
11 8D 06 17	STA 1706 <i>set time to 0.5ms</i>	28 8D 00 17	STA 1700
14 2C 07 17	④ BIT 1707 <i>check time</i>	2B 4C 0F 00	JMP ③ <i>continue</i>

Use Of the ST key for Starting a Program

Cass R. Levart

If you store the starting address of your program in the locations 17FA and 17FB then you can always restart the program by simply pressing the ST key without having to press AD followed by the starting address, followed by pressing GO. For example Hunt the Wumpus starts at 300. You should store 00 in 17FA and 03 in 17FB, to restart the program you then only press ST.

A PARTIAL KIM-1 BIBLIOGRAPHY

FROM RONALD KUSHNIGER
3108 ADDISON COURT
CORNWALLS HEIGHTS, PA. 19008

<u>MAGAZINE</u>	<u>DATE</u>	<u>TITLE</u>	<u>PAGE</u>
BYTE	NOV 1975	SON OF ANTELOPE	66
BYTE	MAY 1976	A DATE WITH KIM	8
MICRO TREK	AUG 1976	KIM-1 COVER STORY	7
BYTE	AUG 1976	HOW I RELATE TO KIM	44
BYTE	SEPT 1976	KIM ON, NOW (LETTER)	93
BYTE	OCT 1976	NEXT OF KIM (LETTER)	136

HERES A HANDY MOVE

ROUTINE FROM →

Edward J. Bechtel, M.D.
351 Hospital Road, Ste 210
Newport Beach, CA. 92663

The MOVE-A-BLOCK program will move a block of bytes up to 256 bytes long forwards or backwards any distance. The block can be across page boundaries -- it does not have to reside in one page. The starting address and ending address of the block is entered in 0000 - 0003. The NEW starting address of the moved block (i.e., where you want to move it) is entered at 0004-5. I located it in 1780 to be generally out of the way, but if you wish, you can use it to relocate itself anywhere.

The program calculates whether the move is forwards or backwards, then moves from the top up, or from the bottom down. The number of spaces the block is moved (in signed notation) is stored by the program in 0006-7, and the number of bytes that were moved is stored in 0008. Also, the new ending address of the moved block is automatically placed in 0002-3, for subsequent use.

```

MOVE-A-BLOCK

1780 38          SEC
1781 A5 04      LDA ZPg 04
1783 E5 00      SBC " 00
1785 85 06      STA " 06
1787 A5 05      LDA " 05
1789 E5 01      SBC " 01
178B 85 07      STA " 07
178D 90 18      BCC +24 MOVEB
178F 38          SEC
1790 A5 02      LDA ZPg 02
1792 E5 00      SBC " 00
1794 A8          TAY
1795 34 08      STY " 08
1797 E6 08      INC " 08
1799 B1 00      LOOP1 LDA (00),Y
179B 91 04      STA (04),Y
179D 88          DEY
179E D0 F9      BNE LOOP1
17A0 B1 00      LDA (00),Y
17A2 91 04      STA (04),Y
17A4 88          DEY
17A5 30 14      BMI END
17A7 38          SEC
17A8 A5 02      LDA ZPg 02
17AA E5 00      SBC " 00
17AC 85 08      STA " 08
17AE E6 08      INC " 08
17B0 A0 00      LDY #00
17B2 B1 00      LOOP2 LDA (00),Y
17B4 91 04      STA (04),Y
17B6 C8          INY
17B7 C4 08      CFY ZPg 08
17B9 D0 F7      BNE LOOP2
17BB 18          END
17BC A5 02      LDA ZPg 02
17BE 65 06      ADC " 06
17C0 85 02      STA " 02
17C2 A5 03      LDA " 03
17C4 65 07      ADC " 07
17C6 85 03      STA " 03
17C8 4C 4E 1C   JMP START

```

```

0000 = SAL)
0001 = SAH) Original
      ) Block of
0002 = EAL) bytes
0003 = EAH)

0004 = SAL) New location
0005 = SAH)

0006 = dif L) Number of spaces
0007 = dif H) block is moved
      (signed notation)
0008 = Number of bytes in block

```

"MOVE-A-BLOCK" works fine! (the editor)

FOR YOU CHESSPLAYERS!!

Chess Clock Program

Cass R. Levart
12 Georjean Drive
Holmdel, N.J. 07733

Program starts at location 200. Two independent clocks are operated by the two players by depressing 1 or 2 respectively. The right two digits show the move number, the left four digits show minutes and seconds. Maximum time is 99 minutes 59 sec. The clock program can be finely tuned by changing the value of word 27F, increase by 1 slows the clock by approx. 6sec/24 hours and vice versa. The value shown of BF was about the best with my unit.

LOCATION	CODE	MNEMONIC	LOC	CODE	MNEMONIC
200	A9 00	LDA #00	23D	85 D3	STA D3
202	AA	TAX	23F	A5 D0	LDA D0
203	9D 00 00 ①	STA,x	241	85 FB	STA FB
206	E8	INX	243	A5 D1	LDA D1
207	D0 FA	BNE ①	245	85 FA	STA FA
209	20 1F 1F ②	JSR DISPL	247	60	RTN
20C	20 6A 1F	JSR GETKEY	248	A5 FB ④	LDA FB
20F	C9 02	CHP #2	24A	85 D0	STA D0
211	D0 F6	BNE ②	24C	A5 FA	LDA FA
213	A9 01 ③	LDA #01	24E	85 D1	STA D1
215	85 D4	STA FLAG	250	A5 D2	LDA D2
217	20 60 02	JSR SUPER	252	85 FB	STA FB
21A	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
21D	A9 02	LDA #02	256	85 FA	STA FA
21F	85 D4	STA FLAG	258	60	RTN
221	20 60 02	JSR SUPER	259-25F	not used	
224	18	CLC	260	F8 ⑩	SED SUPER
225	A5 F9	LDA F9	261	A9 04	LDA #04 SET MULT
227	69 01	ADC #1	263	85 D5	STA D5
229	85 F9	STA F9	265	A9 F0 ⑦	LDA #F0 SET TIME
22B	20 31 02	JSR TRANSF	267	8D 07 17	STA 1707
22E	4C 13 02	JMP ③	26A	20 1F 1F ⑥	JSR DISPL
231	A9 02	LDA #02 TRANSF	26D	20 6A 1F	JSR GETKEY
233	C5 D4	CHP FLAG	270	C5 D4	CHP FLAG
235	D0 11	BNE ④	272	D0 01	BNE ⑤
237	A5 FB	LDA FB	274	60	RTN
239	85 D2	STA D2	275	2C 07 17 ⑤	BIT 1707
23B	A5 FA	LDA FA	278	10 F0	BPL ⑥

Chess Clock Program (cont.)

LOCATION	CODE	MNEMONIC
27A	C6 D5	DEC MULT
27C	D0 E7	BNE ⑦
27E	A9 BF	LDA #BF
280	8D 06 17	STA 1706
283	2C 07 17 ⑧	BIT 1707
286	10 FB	BPL ⑧
288	18	CLC
289	A5 FA	LDA FA
28B	69 01	ADC #1
28D	85 FA	STA FA
28F	C9 60	CHP #60
291	D0 05	BNE ⑨
293	38	SEC
294	A9 00	LDA #00
296	85 FA	STA FA
298	A5 FB ⑨	LDA FB
29A	69 00	ADC #0
29C	85 FB	STA FB
29E	4C 60 02	JMP ⑩

PAGE 7

A REAL "REAL TIME CLOCK" from

Charles H. Parsons
80 Longview Rd.
Monroe, Conn. 06468

I'm really glad that MOS put the timer in the KIM-1 module. I now have a real time clock running off the timer in the interrupt mode. In reading Jim Butterfields' suggestions I felt the easiest way to do this would be to repeatedly enter F4 into the timer each time the interrupt (NMI) occurs. This theoretically produces a time of 249,856 microseconds or just under $\frac{1}{4}$ second. The adjustment to $\frac{1}{4}$ second is done with the same timer in the interrupt program. A fine adjustment of the clock can be made by modifying line 0366. I have added a number of subroutines which use the clock information but I will document only three things here.

1. Real time clock
2. Display clock on the Kim-1 readout
3. Escape to Kim if #1 key on Kim is pressed

The escape to KIM allows KIM to be run without stopping the clock. An exception to this is anything using the NMI such as single step operation. This is a price paid for giving the clock first priority. I also have a speaker hooked to PBO to provide various alarms and sounds. The KIM runs fine in spite of the interrupts but I suspect they would interfere with the audio tape operation. Pressing the KIM GO button will get you out of the KIM loop. Don't forget to connect expansion connector pin 6 to application connector pin 15 per application note #2 !

0080	QSEC	$\frac{1}{4}$ Second Counter
0081	SEC	Second Counter
0082	MIN	Minute Counter
0083	HR	Hour Counter
0084	DAY	Day Counter For AM-PM
17FA	60	NMI Interrupt
17FF	03	Pointers

Interrupt Routine

0360	48	PHA	Save A
0361	8A	TXA	
0362	48	PHA	Save X
0363	9C	TYA	
0364	48	PHA	Save Y
0365	A983	LDA #383	Fine Adjust Timing
0367	8D0417	STA TIME4	
036A	2C0717	BIT TIME7	Test Timer
036D	10FB	BPL TM	Loop Until Time Out
036F	E680	INC QSEC	Count $\frac{1}{4}$ Seconds
0371	A904	LDA #304	Do Four Times Before
0373	C580	CMP QSEC	Updating Seconds
0375	D038	BNE RTN	
0377	A900	LDA #300	Reset $\frac{1}{4}$ Second Counter
0379	8580	STA QSEC	
037B	18	CLC	
037C	F8	SED	Advance Clock In Decimal
037D	A581	LDA SEC	
037F	6901	ADC #301	Advance Seconds
0381	8581	STA SEC	
0383	C960	CMP #360	Until 60 Seconds
0385	D028	BNE RTN	
0387	A900	LDA #300	Then Start Again
0389	8581	STA SEC	
038B	A582	LDA MIN	

```

038D 18      CLC
038E 6901   ADC #01      And Advance Minutes
0390 8582   STA MIN
0392 C960   CMP #060     Until 60 Minutes
0394 D019   BNE RTN
0396 A900   LDA #000     Then Start Again
0398 8582   STA MIN
039A A583   LDA HR       And Advance Hours
039C 18      CLC
039D 6901   ADC #01
039F 8583   STA HR
03A1 C912   CMP #012     Until 12 Hours
03A3 D002   BNE TH
03A5 E684   INC DAY      Advance 1/2 Day
03A7 C913   TH CMP #013  If 13 Hours
03A9 D004   PNE RTN      Start Again With One
03AB A901   LDA #001
03AD 8583   STA HR
03AF D8     RTN CLD      Go Back To Hex Mode
03B0 A9F4   LDA #0F4     Start Timer With Interrupt
03B2 8D0F17 STA TIMEF In 249,856 Microseconds
03B5 68     PLA
03B6 A8     TAY          Restore Y
03B7 68     PLA
03B8 AA     TAX          Restore X
03B9 68     PLA          Restore A
03BA 40     RTI          Return From Interrupt

```

This routine uses the NMI to update a clock in zero page locations. Since the crystal may be slightly off one MHz a fine adjustment is located at 0366. NMI pointers must be set to the start of this program.

Display Clock On KIM-1 Readout

Line	Code	Label	Instruction	Comment
03C0	A900		LDA #000	Reset 1/2 Second Counter
03C2	8580		STA QSEC	
03C4	A9F4		LDA #0F4	Start Timer With Interrupt
03C6	8D0F17		STA TIMEF	
03C9	A581	DSP	LDA SEC	Start Here If Clock Is Running
03CB	85F9		STA INH	Display Clock On KIM
03CD	A582		LDA MIN	
03CF	85FA		STA POINTL	
03D1	A583		LDA HR	
03D3	85FB		STA POINTH	
03D5	201F1F		JSR SCANDS	
03D8	200003		JSR KIM	Escape To KIM
03DB	200002		JSR MTIME	Minute Timer
03DE	202003		JSR REEP	Sound On The Hour
03E1	209002		JSR UPDATE	Calendar
03E4	207502		JSR DSPDAY	Show Date
03E7				
03EA				
03ED				
03F0				
03F3				
03F6				
03F9				
03FC	4CC903		JMP DSP	

PUT EA's (NOP) IN LOC. 03DB-03FB UNTIL
 AND OTHER ROUTINES ARE ADDED. ^{SOME} ADD-
 ITIONAL ROUTINES WILL BE IN AN UPCOMING
 ISSUE - ECR

HELP! Desperately looking for a BASIC Interpreter to run on my KIM-1 System. Will gladly pay! At your mercy!

Edward L. Pavla
 127 Sugar Maple Drive
 Rochester, N. Y. 14615

Escape to KIM if 1 on KIM is Pressed

Line	Code	Label	Instruction	Comment
0300	206A1F	KIM	JSR GETKEY	Go Back To KIM If
0303	C901		CMP #301	KIM Keyboard Is One
0305	D00D		BNE ENDR	
0307	201F1F		JSR SCANDS	Delay To Make Sure
030A	206A1F		JSR GETKEY	
030D	C901		CMP #301	
030F	D003		BNE ENDR	
0311	4C051C		JMP SAVE1	
0314	60	ENDR	RTN	

This is a subroutine which will return to the KIM monitor routine without stopping the real time clock. It is done by pressing 1 on the KIM keyboard.

editors note:

THIS IS BUT ONE METHOD OF SETTING UP A REAL-TIME CLOCK FOR YOUR SYSTEM. ANOTHER WAY TO GO ABOUT WOULD BE TO USE A CLOCK CHIP (SUCH AS THE MM5312 OR MM5313) THAT HAS BCD AND 1 PULSE/SECOND OUTPUT. ONE 8-BIT INPUT PORT WITH INTERRUPT CAPABILITY WOULD DO THE JOB (INTEL 8212?) HAS ANYONE DONE THIS YET???

How Bout TOUCH-TONE ?

A CHIP THAT LOOKS GOOD FOR THIS APPLICATION IS THE MOSTEK MK508CN. IT CAN BE DRIVEN DIRECT FROM ONE 8-BIT OUTPUT PORT AND NEEDS AN INEXPENSIVE COLOR TV XTAL (3.58 MHZ). THE MK508CN IS AVAILABLE FOR \$8.95 FROM TRI-TEK, 6522 N. 43RD AVENUE, Glendale, ARIZONA 85301

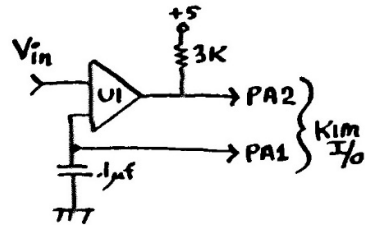
6502 OP CODE TABLE

SYD ALLEN
507 Hill St. #5
Santa Monica, CA 90405

	0	1	2	4	5	6	8	9	A	C	D	E
0	BRK imp	ORA i,x			ORA zer	ASL zer	PHP imp	ORA imm	ASL acc		ORA abs	ASL abs
1	RPL rel	ORA i,y			ORA spx	ASL zpx	CLC imp	ORA aby			ORA abx	ASL abx
2	JSR abs	AND i,x		BIT zer	AND zer	ROL zer	PLP imp	AND imm	ROL acc	BIT abs	AND abs	ROL abs
3	EMI rel	AND i,y			AND spx	ROL zpx	SEC imp	AND aby			AND abx	ROL abx
4	RTI imp	EOR i,x			EOR zer	LSR zer	PHA imp	EOR imm	LSR acc	JMP abs	EOR abs	LSR abs
5	BVC rel	EOR i,y			EOR spx	LSR zpx	CLI imp	EOR aby			EOR abx	LSR abx
6	RTS imp	ADC i,x			ADC zer		PLA imp	ADC imm		JMP ind	ADC abs	
7	BVS rel	ADC i,y			ADC spx		SEI imp	ADC aby			ADC abx	
8		STA i,x		STY zer	STA zer	STX zer	DEY imp		TXA imp	STY abs	STA abs	STX abs
9	BCC rel	STA i,y		STY spx	STA spx	STX zpy		TYA imp	TXS imp		STA abx	
A	LDY imm	LDA i,x	LDX imm	LDY zer	LDA zer	LDX zer	TAY imp	LDA imm	TAX imp	LDY abs	LDA abs	LDX abs
B	BCL rel	LDA i,y		LDY spx	LDA spx	LDX zpy		CLV imp	LDA aby	TSX imp	LDY abx	LDX aby
C	CPY imm	CMP i,x		CPY zer	CMP zer	DEC zer	INY imp	CMP imm	DEX imp	CPY abs	CMP abs	DEC abs
D	BNE rel	CMP i,y			CMP spx	DEC spx	CLD imp	CMP aby			CMP abx	DEC abx
E	CPX imm	SBC i,x		CPX zer	SBC zer	INC zer	INX imp	SBC imm	NOP imp	CPX abs	SBC abs	INC abs
F	BEQ rel	SBC i,y			SBC spx	INC spx	SED imp	SBC aby			SBC abx	INC abx

abs Absolute
abx absolute indexed using x register
acc accumulator
aby absolute indexed using y register
i,x indexed indirect using x register
i,y indirect indexed using y register
imm immediate
imp implied
ind indirect
rel relative
zer zero page
spx zero page indexed using x register
spx zero page indexed using y register

least sig 4 bits
n
s
t
s
i
s
t
4
b
i
t
n



$V_{in} = >.25v$ and $<4.0v$

UI = LM311

IDEA FOR SOFTWARE DRIVER

① PROGRAM PA1 AS OUTPUT, PA2 AS INPUT, WRITE "0" TO PA1

② LOAD TIMER WITH "FF"

③ WRITE "1" TO PA1

④ LOOP 'TIL PA2 GOES HIGH.

⑤ READ THE TIMER + SUBTRACT READING FROM "FF"

⑥ WRITE "0" TO PA1 TO LET CAPACITOR DISCHARGE.

⑦ JUMP BACK TO STEP 2

LOW-COST A/D
by Rick Simpson
(reprinted from
Kim User Notes
COMPLEMENTARY
ISSUE)

TAPE DUPE

Jim Butterfield, Toronto

The best way to copy a program to another tape is to read it in, then write it out. This completely regenerates the level, waveform, frequencies and timing of the tape.

If you have a lot of programs to copy, doing this manually becomes a tedious business. With a little hardware to connect to the remote control jacks of the cassette recorders, you could generate an automatic copier program. The tapes would start and stop under program control. Challenge: who's going to be the first to submit such a program to USER NOTES?

In the meantime, here's a little program to copy all the contents of one tape to another. It regenerates the level, waveform, and frequencies, but not the timing. Three out of four isn't bad. It can't quite manage Supertape, but all other speeds--regular, 2x and 3x--will copy OK.

Connect your two cassette recorders in the usual way, at the A.DIG IN and A.DIG OUT points. With the program running, start the recorders. All programs will be copied from one tape to the other.

Program TAPE DUPE is fully relocatable.

```
0000 A9 27 START LDA #27 SED value
0002 A2 3F GO LDX #3F Directional reg's
0004 EE 43 17 STX PHED set for input
0007 A2 07 LDX #7 PBS (CONT) set
0009 EE 42 17 STX SED for input
000C A0 5E LDY #94 High frequency
000E 2C 42 17 HIT SED Zero or one?
0011 10 02 BPL OVER
0013 A0 A3 LDY #163 Low frequency
0015 A2 1F OVER LDX #1F Directional reg's
0017 EE 43 17 STX PHED set for output
001A 49 80 EOR #80 Reverse output bit
001C ED 42 17 STA SED and send it
001F 8C 44 17 STY CLKLT Set timer
0022 2C 47 17 WAIT HIT CLKRDI ..and wait
0025 10 FB BPL WAIT
0027 30 D9 BMI GO
```

T. W. Hubbell
533 Wintergreen Cr.
Victor, N.Y. 14564

Keep up the good work.

Here is a short program that may be of interest to others. I wanted to load from tape under program control using the KIM load memory from tape at 1871 but had some difficulty returning from the monitor. I use the KIM recommended speaker interface driving a 15ma 6v relay to turn the recorder on and off from PB-2. I finally found I could break in at "VEB" 17EC.

LOAD TAPE BREAK -

Set up the following:

```
17FE 5800 set up vector
00E6 XX reserved to save accumulator
00E7 ELEH tape , end low, end high, address
00E9 8DXXX emulates "VEB"
00EC 4CEP17 JMP VEB + 3
```

```
START -
0050 A900 LDA #00 set brk
0052 8DEC17 STA 17EC
0055 4C7818 JMP 1878 return to KIM
0058 85E6 STA E6 save
005A ADEE17 LDA 17EE SAH
005D 85EB STA EB
005F ADED17 LDA 17ED SAL
0062 85EA STA EA
0064 C5E7 CMP E7 end low
0066 D007 BNE 07 006P
0068 ADEE17 LDA 17EE SAH
006B C5EB CMP EB end hi
006D F005 BEQ 05 0074 EXIT
006F A5E6 LDA E6 get accum.
0071 4CE900 JMP 00E9 jump to "VEB"
0074 4CXXXX JMP Your exit
```

NOTES:
Load tape ID to 17F9 as usual (load from program).
When recording tapes for this use, record to end address + 2, rather than the usual + 1. The end add. above is the + 1. (Otherwise the Monitor will see the end character on the tape before it gets back to BEQ).
Sorry but this will not exit if it is called up, and return, as a sub; perhaps someone can debug that?
Caution the stack will be pushed down in page 1 quite far.

It's not perfect, but it does allow KIM to call up and sort a lot of tape a little at a time, etc.

ROBERT G. LLOYD
7554 Southgate Rd.
Fayetteville, N.C. 28304
(919) 867-5822

Here is a program that I wrote in Pittman Tiny BASIC.

The program lets my children Robin 12 & Bobby 8 play with the computer and at the same time learn math.

I do not have a Teletype so I can't send you a listing of the running program. I am sending a copy of what is on the TVT.

THIS IS A MATH TEST

$$\begin{array}{r} 12 \\ \times 6 \\ \hline \end{array}$$

For the right answer 72 - YOU'RE RIGHT - and a new problem is set up.

For a wrong answer 62 - ?? WRONG ??, TRY AGAIN - the same problem is set up if you get it WRONG 3 times - THE RIGHT ANSWER IS 72

THE PROBLEMS ARE RANDOM, the limits are set at line#s 200 for X & 205 for Y for multiplication & at 305 for X & 355 for Y for addition.

```
10 PR "THIS IS A MATH TEST"
15 PR
20 LET V=0
30 LET I=0
35 LET Z=0
40 PR "TYPE 1 FOR MULTIPLICATION"
50 PR
60 PR "TYPE 2 FOR ADDITION"
70 PR
80 INPUT I
90 PR
100 IF I=1 GOTO 200
110 IF I=2 GOTO 300
120 IF D=Q GOTO 500
130 IF D<>Q GOTO 600
190 END
200 LET X=(RND (12)+1)
205 LET Y=(RND (12)+1)
210 IF X<10 GOTO 230
220 IF X>10 GOTO 240
230 PR "  ",X
235 GOTO 260
240 PR "  ",X
260 IF Y<10 GOTO 280
270 IF Y>10 GOTO 290
280 PR " X ",Y
285 GOTO 300
290 PR " X ",Y
300 PR " ",
310 LET Q=X*Y
320 INPUT D
330 GOTO 120
350 LET X=(RND (50)+1)
355 LET Y=(RND (50)+1)
360 IF X<10 GOTO 380
370 IF X>10 GOTO 390
380 PR "  ",X
385 GOTO 410
390 PR "  ",X
410 IF Y<10 GOTO 430
420 IF Y>10 GOTO 440
430 PR " + ",Y
435 GOTO 450
440 PR " + ",Y
450 PR " ",
460 LET Q=X+Y
470 INPUT D
480 GOTO 120
500 PR "YOU'RE RIGHT"
505 PR
508 LET Z=Z+1
509 IF Z<3 GOTO 512
510 GOTO 100
512 IF I=1 GOTO 200
514 IF I=2 GOTO 300
600 PR " WRONG , TRY AGAIN"
610 PR
620 LET V=V+1
630 IF V=3 GOTO 650
640 IF I=1 GOTO 200
645 IF I=2 GOTO 300
650 PR "THE RIGHT ANSWER IS ",
655 PR Q
660 PR
670 GOTO 100
```

more Kim bibliography
from BRUCE LAND
6916 PARK PLACE
BALTIMORE, MD.
21227

If you could publish a list of Kim-1 and 6502 articles published in other journals, it would be of great value. Can anyone add to this list?

Microtrek; Aug 1976; p. 7-16; "Kim-1 Micro Computer Module"; contains overview of Kim-1, useful executive additions, "drunk test" game.

Interface Age; Nov 1976; p. 103-111; "Floating Point Routines for 6502"; contains good annotated listings, ln, log, exp, +, -, *, x, fixed to floating conversions, loads 1D00-1FEE.

Interface Age; Nov 1976; p. 12-14; "Build a Simple A to D", simple circuit, 6502 software 16 locations, use to interface a pot or joy stick.

Dear Eric:

I am writing to tell you about some of the experiences I have had with Jim Butterfield's "Supertape:" program and its derivatives, "fastape" and "speedtape." I have a number of different models of cassette machines available to me, and I have been primarily using stereo cassette tape decks manufactured by J. V. C. and Craig. When I first attempted to use Jim's programs, I could get "fastape" and "speedtape" to run fine, but the "supertape" program after initial synchronization and the reading of a few Bytes, would become unsynchronized, resulting in an abortive read. Also, the level settings were extremely critical to even get initial synchronization. These observations were made by means of the use of a "VU Tape" program.

After some experimentation with the various values indicated on page 12 of Volume One, Issue Two of KIM-1 User Notes, I have found that loading hex value 03 into address 01BE, and hex value 02 in address location 01C0 seems to give virtually fool-proof read/write performance to my system over an extremely wide range of input levels and types of cassette. I have used Maxell UD, Realistic low noise, and Sony low-noise tape, to mention a few.

One other item of interest concerning supplies for the KIM-1 should be mentioned. Of course, one can never be too careful with ones choice of power supply protection and regulation. The route I have taken is to use an existing well-regulated supply capable of delivering either nine (9) volts or twelve (12) volts at approximately 1.2 amperes. I take the twelve (12) volt output line and feed it into an LM-309-K voltage regulator mounted on a heat sink. The output of the LM-309-K, of course, goes to the five (5) volt buss of the KIM-1, and the twelve (12) volt supply output line also goes to the twelve (12) volt buss which operates the phase-locked loop circuitry on the KIM. One may crowbar the output of the LM-309-K if desired. I have found that by reducing the original power supply output voltage to 9 volts, the LM-309-K operates at greatly reduced heat dissipation requirements, while the phase-locked loop circuitry, operating at the nine (9) volt level, seems

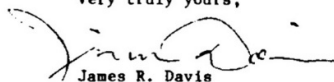
practically unimpaired in performance. This is true even when reading full-speed "supertape:" programs off of tape.

I also want to say that I think the User Notes is a very fine effort, and although I read a great many "slick" micro processor magazines, I know that the User Notes, when it comes, will always have something I can really use.

The single most important thing, from my standpoint, that anyone could come up with for the KIM, would be a software method of teaching KIM to read and write serial baudot, using the resident firmware to shorten such a program as much as possible. The machine should have the capability of operating in the "baudot" mode when running other programs.

Thanks again, Eric, for a most valuable publication.

Very truly yours,


James R. Davis

PAGE 10

IS ANYONE WORKING ON A KIM-1
FLOPPY DISC INTERFACE ?

~ the editor ~

WHAT THE HELL IS TOP-DOWN PROGRAMMING?

Jim Butterfield, Toronto

If you hang around with programming types, you're likely to hear a couple of buzzwords that are popular these days: structured programming; and top-down programming.

The experts don't agree on exactly what the terms mean. Some say that they are a type of computer language; others claim that they are a way of thinking. Read on and make your own opinions.

We'll pass by structured programming rather quickly. It's related to top-down programming techniques. But structured programming doesn't adapt too well to machine language or assembler programming; it doesn't even fit Tiny Basic. So we'll concentrate our efforts on top-down programming, which can indeed be useful to the small computer programmer.

In principle, top-down programming means this: try to avoid your programs jumping about too much. Instead, try to get your program to flow smoothly from the start to the end. (Subroutines are OK, since the program flow always returns to where it left off).

What does that mean in real terms? Let's take some examples.

Suppose we're writing a little division routine. At this point in the program, we have the number to be divided in the accumulator. The divisor, suitably shifted, is in location DVSR, and our task is this: If the accumulator is not less than DVSR, subtract DVSR and add one to QUOT, the quotient. We might be tempted to write:

```

      CMP DVSR          ...elsewhere in the program:
      BCS SUB           SUB SEC
NEXT  ..program continues      SEC DVSR
                                   INC QUOT
                                   JMP NEXT
```

What can we do with this to make it top-down? Well, the problem with the above coding is that we jump out of line to get to SUB, and then have to jump back. (And don't forget that most programming errors are caused by bad Branches and Jumps). A little top-down thinking produces:

```

      CMP DVSR
      BCC NEXT
      SEC DVSR
      INC QUOT
NEXT  .. program continues
```

See how the program 'flows through'? We've saved space, and the coding is easier. (The missing SEC is a gift; the carry's set anyway).

That seems a little too simple. Let's take a slightly tougher one. Somewhere in the program, we need to set the X register either of two ways: to 10 if the accumulator is positive, or to 20 if the accumulator is negative.

Seems like we can't top-down this one. Either the positive accumulator situation or the opposite will have to branch out, it seems. You can't "flow through" and have it both ways, right?

Wrong. Try this:

```

      LDX #10
      TAY          to test accumulator only
      BPL POS      if positive, leave X at 10
      LDX #20      ..else change X to 20
POS     ..coding continues
```


Are you starting to see the idea? Keep that flow in order whenever you can ... you'll end up with easier, short branches; and you'll often save memory!

As a final example: sometimes you can eliminate branches entirely by careful use of the ORA, AND, EOR, and ADC instructions. Often, when you need to generate a flag or special value, you can calculate it rather than testing and branching.

Let's look at the Lunar Landing program previously published in User Notes. This part of the program (which follows a call to KIM routine GETKEY) is testing for the keys A (altitude) or F (fuel) ... (since the program is in decimal, A is 10 and F is 15). We'll assume that keys B, C, D, and E may be allowed to produce the same result as F:

NON-TOP-DOWN CODING	TOP-DOWN CODING
DOKEY CMP #15 F ?	DOKEY CMP #10 numeric?
BNE NAL2	BCC NAL2
STA MODE set alt mode	EOR #10 A becomes 0
RTS	STA MODE 0 or non-zero
NAL2 CMP #10 A?	RTS
BNE NAL2	NAL2 ...continues
LDA #00	
STA MODE	
RET	
NAL2 ECS RET non-numeric?	

See how the EOR eliminates all that testing?

The advantages are obvious. So: next time you're programming, take it from the top!

**NIAGARA
COLLEGE**
OF APPLIED ARTS
& TECHNOLOGY



Woodbine Road
Welland, Ontario
735-2211
1.38.532

We are presently using the KIM-1 systems at the college to teach students in their third year operational, programming and interfacing techniques involved in the use of microcomputers.

If you know of any other educational institution currently using the KIM-1 (or any other 6502 configuration) please let me know.

Yours respectfully,

John W. Clark,
School of Applied Science
and Technology.

maybe all
the educators
should get in
touch (?)

1. I use the fourth letter of the mnemonic to indicate mode. This keeps the source/destination uncluttered. I = immediate, B = absolute, Z = zero page, A = accumulator, "blank" = implied or relative, U = indirect X, V = indirect Y, W = zero page X or Y, X = absolute X, Y = absolute Y, (JMPB & JSRB are used so much, I leave the B off.)

i.e., STAB 1700 8D 00 17 Store A in port "a": 1700
 LDAZ E6 A\$ E6 Load A from 00E6
 ROLX 0300 3E 00 03 Rotate left data stored in 0300+X
 LDXI EF A2 EF X ← EF

2. A "psuedo" BIT immediate uses KIM-1 monitor permanent data, allowing you to search the accumulator for several single bits or bit patterns in succession without first storing the "masks."

i.e., BITB IC53 2C 53 IC Z ← 0 if bit 0 of accumulator is one
 (01 stored in permanent memory IC53)
 other single bit address/data in KIM 1: 1CCB/02, 1CAF/04, 1CC7/08,
 1C99/10, 1C88/20, 1C7F/40, 1A69/80.

Coming up:

More games—
 a software driver for
 the SWTP GRAPHICS Display.

UTILITY PROGRAMS.

A/D converters

WHAT HAVE YOU DONE
 WITH YOUR KIM-1?

How 'BOUT SOME HARDWARE
 STUFF?