

KIM-1/6502 USER NOTES

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

These really files 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 (bb16?), CMOS COOKBOOK, will be reviewed in an upcoming issue of the USER NOTES.

Robert Cushman, Special Features Editor for EM (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 Looftourrow 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 manufacturer 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-Case Lewart informed me that on page 7, the mnemonic in address location 22 should be LDY #77 (not LDX #77), 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 subscribers - \$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 ODTL and ODF2, when you hit 00, the contents of ODTL transfer to the status register, and F2 to the stack pointer. Always preset ODTL to 00 to avoid being accidentally in decimal mode; and ODF2 to F to avoid having the stack "write over" your page 1 programs or data.
.. Jim Butterfield

KIM-1 TO S-100 BUS ADAPTER
Get a flyer from Forthought Products. They announced KIMS1, an 8-bit 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 Forthought 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. Our ve want to be sure no one misses any data. Please contact me if you were shorted one or two back issues recently.....

CV RECEIVE ROUTINE

Z2 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 TVR 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's - THERE IS AN AUTO-START NET ON 80 METERS
(3637.5 KHZ +/- 10 HZ) THAT INCLUDES SOME KIM-1'S. FOR MORE
INFO, CONTACT TRUMAN BOERKOEL K8JUG,
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... On the program with that neat piece of coding in it, that you suddenly need to remove (say, you'll have to fill all subroutines) ... but if you do, you'll have to change to a program 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 up a program to fit in an extra instruction, closing up a space you don't need, or just moving the whole thing somewhere 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 BRL instruction and try to correct the branch address), and place a ddd instruction (FF) behind your last program instruction. This tells RELOCATE where to stop.
- (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.
- (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 FF0C (effectively, a negative number).

Place the boundary address in locations EC and ED, low order first.
 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 FF0C (effectively, a negative number).

Place the increment value in locations E9 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/O 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 initialised
: by user prior to run
PAGLIM ***+1 limit above which KIM reloc
ADJUST ***+2 adjustment distance (signed)
POINT ***+2 start of program
BOUND ***+2 lower boundary for adjustment
: main program starts here
START CDD
LDY #0
LDA (POINT),Y
TAX
LDX #7
LOOP TTA restore op code
AND TABL-1,X remove unwanted bits
BOR TABL-1,X & test the rest
BEQ FOUND
DEI on to the next test
BNE LOOP ... if any
LDY TABL,X length or flag
BEQ TRAP triple length
BRN BRAN branch!
INC POINT M-ang right along..
BNE INEX .. to next op code
INC POINT+1
DEI
BNE SKIP
BEQ START
: length 3 or illegal
TRIP INT
INT INT
BNE START+2 illegal/end to BRK halt
INT INT
LDA (POINT),Y Io-order operand
TAX ... into X reg
INT

```

I REALLY FLIPPED WHEN I SAW THIS PROGRAM BUT I GOT OVER IT AGAIN !! FOR

```

013E B1 EA LDA (POINT),Y hi-order operand
0140 20 79 01 JSR ADJUST change address, maybe
0143 91 1A STA (POINT),Y ..and put it back
0145 88 DEI Y=1
0146 8A TTA
0147 91 5A STA (POINT),Y ..also hi-order
0149 40 03 LDY #3 Y=3
014B 10 DE BRL SKIP
014D 0E BHEM : branch: check 'to' and 'from' addresses
014F A6 EA INY Y=1
0150 A5 EB LDA POINT+1 'from' addr Io-order
0152 20 79 01 JSR ADJUST .. & hi-order
0155 86 E0 SIT ALOC change, maybe
0157 A2 FF LDY #FF save Io-order only
0159 B1 EA LDA (POINT),Y flag for 'back' branches
015B 18 CLC
015C 69 02 ADC #2 adjust the offset
015E 30 01 BPT OVER backwards branch
0160 B8 INX nope
0161 86 E3 OVER
0163 65 EA CLC calculate 'to' Io-order
0164 65 EA ADC POINT .. and put in X
0166 AA TAX
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
0170 BA DEX
0171 38 TTA
0172 E5 E0 SEC
0174 91 EA STA (POINT),Y recalculate relative branch
0176 C8 INY and re-incremt
0177 10 B2 BRL SKIP Y=2
0179 C5 F7 : examine address and adjust, maybe
017B B0 11 ADJUST CDP PAGLIM
017C C5 ED RES OUT too high
017F D0 02 BNE TEST2 high-order!
0181 E4 EC CYP BOUND Io-order!
0183 90 09 BEC OUT too low!
0185 48 PNA stack hi-order
0187 BA TTA
0188 65 F8 ADC ADJUST adjust Io-order
018A AA TAX unstack hi-order
018B 68 PNA end adjust
018C 65 E9 ADC ADJUST+1
018E 60 RTS
018F 0C 1F 0D TAB1 :tablas for op-code identification
0192 E7 1F FF .BITS $OC,$1F,$0D,$E7,$1F,$FF,$03
0195 09 .BITS $0C,$19,$08,$00,$10,$20,$03
0196 0C 19 08 TAB2
0199 00 10 20 .BITS $0C,$19,$08,$00,$10,$20,$03
019C 03 02 FF FF TAB3
019D 01 01 00 .BITS $0C,$1F,$0D,$E7,$1F,$FF,$03
01A0 01 01 00 .BITS $0C,$1F,$0D,$E7,$1F,$FF,$03
01A3 FF FF
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 !!

KIM-1 - 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 IX means the 'from' address minus one; 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 BD IX IX 9D TT TT CA DO P7 00

Example: move contents of 0234-0278 to 0258-0284

A2 45 BD 13 02 9D 12 02 CA DO P7 00

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

A2 00 B8 BD IX IX 9D TT TT BD NN DO P5 00

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

A2 00 B8 BD 57 02 9D 33 02 E0 31 DO P5 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 P7 into location 1779 and perform a tape read.

Tom Wear
380 Belaire
Punta Gorda, FL33950

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 W4 74145 for the 'IS' 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, DIRECTORY

Jim 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?

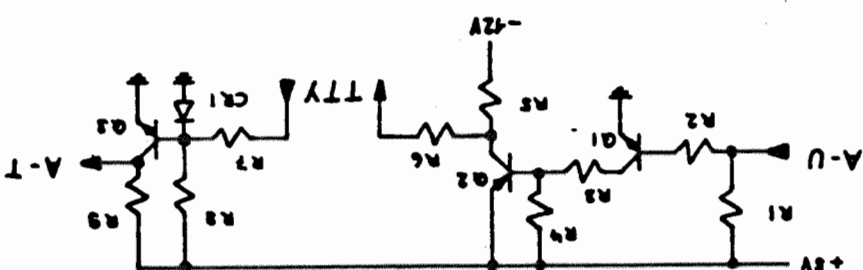
I suggest the following: First digit - 0 to 9 for completed (or 'permanent') programs ... A to F for programs you're still working on! Second digit - 0 to 9 for programs, A to F for data files. Using this scheme, I'd know that ID 5E is a permanent data file! A3 is a program still being writ.

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 195C 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	C1D	Directional reg
0001 A9 07		S7A SHD	
0003 8D 42 17		JSR RBBIT	Scan thru bits...
0006 20 41 1A SYN		ISR INH	..shifting new bit
0009 46 F9		ORA INH	..into left of
000B 05 F9		S7A INH	..byte INH
000D 85 F9		CMP #16	SYNC character?
000F C9 16	TST	BNE SYN	no, back to bits
0011 DO F3		JSR RDOHT	get a character
0013 20 24 1A		DEC INH	count 22 SYNC's
0016 C6 F9		BPL TST	
0018 10 F5		CMP #32A	then test astk
001A C9 2A		BNE TST	..or SYNC
001C DO F1		LDX #8FD	if asterisk,
001E A2 FD		JSR RDBYT	stack 3 bytes
0020 20 F3 19 RD		S7A POINTH+1,X	into display
0023 95 FC		INX	area
0025 E8		BMI RD	
0026 30 F8		JSR SCANDS	...and shine
0028 20 1F 1F SHOW		BNE GO	until keyed
002B DO DJ		BEG SHOW	at's all folks
002D F0 F9			



KERES AN RS-232 INTERFACE FROM: Markus P. Goerner
Hertlbergstr. 107
8045 Zurich
Schweiz

Parts list:

CR 1	2 N 2222
CR 2	2 N 2907
CR 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

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?

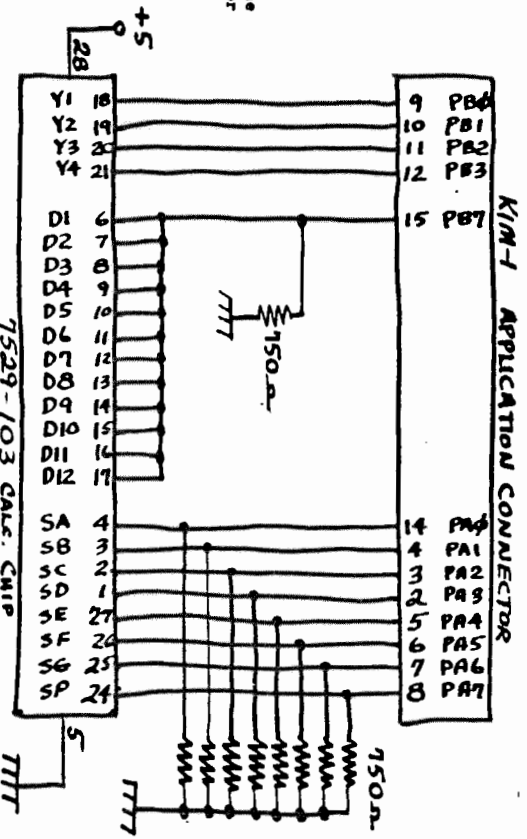
One example of the circuitry necessary to interface the 7529-103 to a micro was presented in Byte (Sept, Oct 1976). This circuit used about 29 IC's to get a two way conversation going with the calculator chip. That's more IC's than there are on Kim! There has to be a better way.

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.

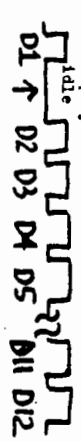
There is one hardware "trick" that you should be aware of: originally, the 7529-103 was designed to work with a negative 7.5v supply. If you saw the Byte article, you can see how the chips operating point can be shifted up to use a positive 7.5 volt supply (just reverse the Ground and Vdd connections). 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 keyboard 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. (It 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 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....



KEY CODE	KEY CODE	KEY CODE	KEY CODE
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
ARC	ARC	ARC	ARC
C2	C2	C2	C2
C8	C8	C8	C8
1/X	1/X	1/X	1/X
x ²	x ²	x ²	x ²
10 ^x	10 ^x	10 ^x	10 ^x
x ³	x ³	x ³	x ³
x ⁴	x ⁴	x ⁴	x ⁴
x ⁵	x ⁵	x ⁵	x ⁵
x ⁶	x ⁶	x ⁶	x ⁶
x ⁷	x ⁷	x ⁷	x ⁷
x ⁸	x ⁸	x ⁸	x ⁸
x ⁹	x ⁹	x ⁹	x ⁹
N1	N1	N1	N1
48	48	48	48
82	82	82	82
92	92	92	92
CHS	CHS	CHS	CHS
A2	A2	A2	A2
EXX	EXX	EXX	EXX
B2	B2	B2	B2
04	04	04	04
SIN	SIN	SIN	SIN

CALCULATOR DRIVER

0000	*12 Data Output File	0200	49 0F	init
000C	*12 Temp Data Buffer	0202	8D 03 17	
0018	Keystroke (Temp)	0205	A2 00	exec
0019	X Store (Temp)	0207	BD 00 03	ex 1
1700	PA Data	020A	86 19	
1702	PB Data	020C	20 23 02	
1703	PB Control Reg.	020F	A6 19	
		0211	A5 18	
		0213	C9 FF	
		0215	DC 03	
		0217	4C 8C 02	
		021A	B8	more
		021B	BD 03	
		021D	4C 07 02	
		0220	20 05 1C	noFF
		0223	85 18	calc
		0225	A0 04	
		0227	2C 02 17	A1
		022A	30 FB	
		022C	A2 14	
		022E	C0 FD	A2
		022F	DC 02 17	
		0234	30 F1	
		0236	A5 18	
		0238	C9 FF	
		023A	FD 34	
		023C	4A	
		023E	4A	
		023F	4A	
		0240	AA	
		0241	CA	A3
		0242	FC 02	
		0244	2C 02 17	A4
		0247	10 FB	
		0249	2C 02 17	A5
		024C	30 FB	
		024E	10 F1	
		0250	A5 18	write
		0252	29 0F	
		0254	AA	
		0255	2C 02 17	B1
		0258	10 FB	
		025A	8E 02 17	
		025D	A2 00	
		025F	2C 02 17	B2
		0262	30 FB	
		0264	8E 02 17	
		0267	88	
		0268	DO BD	
		026A	20 A0 02	B3
		026D	EA	
		026F	EA	
		0270	60	
		027E	EA	B4
		028F	60	

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

continued on next page

0270	A0 08	read	LDI #0B	difer-1
0272	A2 14	20	LDI 14	walt 100 wsee
0274	2C 02 17	C1	BIT PBD	high synch?
0277	10 FB	C2	BPL C1	not yet?
0279	CA CA		DEX	
027A	DO FD		BNE C2	
027C	AD 00 17		LDA PAD	read calc. output
027F	99 0C 00		STA	store code
0282	88		DEX	
0283	30 EA		BMI B4	
0285	2C 02 17	C3	BIT PBD	low synch?
0288	30 FB		BPL C3	
028A	10 E6		BPL C0	
028C	AD 01		LDI 01	rearrange
028E	A2 04		LDX 04	digits
0290	B5 00		LDA 0000,1	to
0292	99 00 00		STA 0000,1	proper
0295	C8		INY	order...
0296	CA		DEX	
0297	10 F7		BPL move	
0299	A5 0C		LDA	
029B	85 00		STA	
02A0	4C 4F 1C	delay	JMP back to KIM	
02A2	8D 05 17	walt	LDA #2C	out up time delay
02A5	2C 07 17		STA CLKRT +8	
02A8	10 FB		BIT CLKRT	time out?
02AA	2C 00 17	B3B	BPL walt	
02AD	10 FB		BIT ADRT	look for high
02AF	60		BPL B3B	segment P
			RFS	back to calo

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
17F1 4C 0F 19         jmp LOAD12
17F4 4C 29 19         failed jmp LOAD19
      
```
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 \$FFF.

I thoroughly enjoyed HUNT THE WUMPIUS 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.

WANTED: ANY DATA ON CONSTRUCTING A SIMPLE TVT FROM SCARTCH - DAN GARBER 11825 BEACH BLVD STAMFORD CAL 90680

Interface for the SouthWest Technical Products TV typewriter II and KIM-1. The SWTP serial interface board is used. Jumper between terminals V and Z1 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

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.

Notes: 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
01 A9 15	LDA#15	19 A5 2E
03 85 2E	STA 2E	1B FO E4
05 A9 FF	LDA #FF	1D C6 2E
07 8D 01 17	STA 1701	1F 20 6A 1F
0A A9 01	LDA #01	22 C5 2E
0C 8D 00 17	STA 1700	24 DO E9
0F A9 05	LDA #05	26 A9 00
11 8D 06 17	STA 1706	28 8D 00 17
14 2C 07 17	BIT 1707	2B 4C 0F 00

BPL ①	LDA 2E
BEQ ②	clear led's & rings
DEC 2E	dec. volume at counter
JSR GETKEY	get key & store in counter
CMP 2E	compare counter & key
BNE ③	if not equal
LDA #00	set Pao (triac) led's
STA 1700	to 0
JMP ④	continue

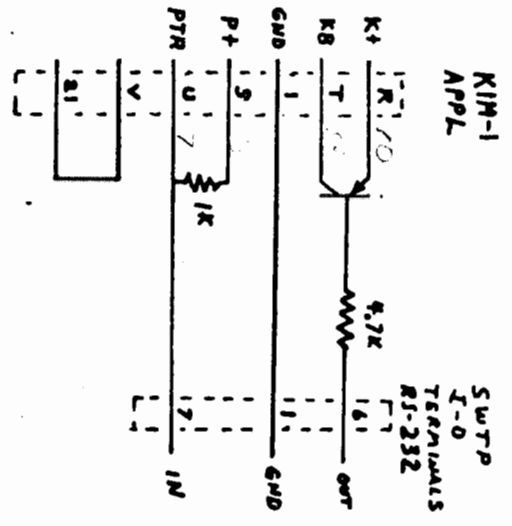


Case R. Lewart
133 Geo. Jean Dr.
Holmdel, N.J. 07733

Use Of the ST key for Starting a Program

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.

Case R. Lewart



FOR YOU CHESSPLAYERS !!

Chess Clock Program
 Case R. Lovart
 12 Georgian 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 27E, 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.

Chess Clock Program (cont.)

PAGE 7

A PARTIAL KIM-1 BIBLIOGRAPHY FROM
 RONALD SUMNER
 3100 ABBOTT COURT
 CONNELL MOUNTAIN, PA. 15808

DATE	TITLE	PAGE
NOV 1975	SOUL OF MATHS	56
MAY 1976	ADVICE WITH KIM	8
AUG 1976	KIM-1 COVER STORY	7
AUG 1976	MICRO I RETURN TO KIM	44
SEPT 1976	KIM ON MATHS (PART 1)	43
OCT 1976	NEXT OF KIM (PART 1)	136

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

ROUTINE FROM →
 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.

LOCATION	CODE	NMNEMONIC	LOC	CODE	NMNEMONIC
200	A9 00	LDA #00	23D	85 D3	STA D3
201	AA	TAX	23E	A5 D0	LDA D0
202	9D 00 00	STA,X	241	85 FB	STA FB
203	EB	INX	243	A5 D1	LDA D1
204	DO FA	BNE ①	245	85 FA	STA FA
205	20 1F 1F	JSR DISPL	247	60	RTN
206	20 6A 1F	JSR GETKEY	248	A5 FB	LDA FB
207	C9 02	CMP #2	24A	85 D0	STA D0
208	DO F6	BNE ②	24C	A5 FA	LDA FA
209	A9 01	LDA #01	24E	85 D1	STA D1
210	85 D4	STA FLAG	250	A5 D2	LDA D2
211	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
212	A9 02	LDA #02	256	85 FA	STA FA
213	85 D4	STA FLAG	258	60	RTN
214	20 60 02	JSR SUPER	252	85 FB	STA FB
215	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
216	A9 02	LDA #02	256	85 FA	STA FA
217	85 D4	STA FLAG	258	60	RTN
218	20 60 02	JSR SUPER	252	85 FB	STA FB
219	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
220	A9 02	LDA #02	256	85 FA	STA FA
221	85 D4	STA FLAG	258	60	RTN
222	20 60 02	JSR SUPER	252	85 FB	STA FB
223	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
224	A9 02	LDA #02	256	85 FA	STA FA
225	85 D4	STA FLAG	258	60	RTN
226	20 60 02	JSR SUPER	252	85 FB	STA FB
227	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
228	A9 02	LDA #02	256	85 FA	STA FA
229	85 D4	STA FLAG	258	60	RTN
230	20 60 02	JSR SUPER	252	85 FB	STA FB
231	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
232	A9 02	LDA #02	256	85 FA	STA FA
233	85 D4	STA FLAG	258	60	RTN
234	20 60 02	JSR SUPER	252	85 FB	STA FB
235	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
236	A9 02	LDA #02	256	85 FA	STA FA
237	85 D4	STA FLAG	258	60	RTN
238	20 60 02	JSR SUPER	252	85 FB	STA FB
239	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
240	A9 02	LDA #02	256	85 FA	STA FA
241	85 D4	STA FLAG	258	60	RTN
242	20 60 02	JSR SUPER	252	85 FB	STA FB
243	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
244	A9 02	LDA #02	256	85 FA	STA FA
245	85 D4	STA FLAG	258	60	RTN
246	20 60 02	JSR SUPER	252	85 FB	STA FB
247	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
248	A9 02	LDA #02	256	85 FA	STA FA
249	85 D4	STA FLAG	258	60	RTN
250	20 60 02	JSR SUPER	252	85 FB	STA FB
251	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
252	A9 02	LDA #02	256	85 FA	STA FA
253	85 D4	STA FLAG	258	60	RTN
254	20 60 02	JSR SUPER	252	85 FB	STA FB
255	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
256	A9 02	LDA #02	256	85 FA	STA FA
257	85 D4	STA FLAG	258	60	RTN
258	20 60 02	JSR SUPER	252	85 FB	STA FB
259	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
260	A9 02	LDA #02	256	85 FA	STA FA
261	85 D4	STA FLAG	258	60	RTN
262	20 60 02	JSR SUPER	252	85 FB	STA FB
263	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
264	A9 02	LDA #02	256	85 FA	STA FA
265	85 D4	STA FLAG	258	60	RTN
266	20 60 02	JSR SUPER	252	85 FB	STA FB
267	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
268	A9 02	LDA #02	256	85 FA	STA FA
269	85 D4	STA FLAG	258	60	RTN
270	20 60 02	JSR SUPER	252	85 FB	STA FB
271	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
272	A9 02	LDA #02	256	85 FA	STA FA
273	85 D4	STA FLAG	258	60	RTN
274	20 60 02	JSR SUPER	252	85 FB	STA FB
275	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
276	A9 02	LDA #02	256	85 FA	STA FA
277	85 D4	STA FLAG	258	60	RTN
278	20 60 02	JSR SUPER	252	85 FB	STA FB
279	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
280	A9 02	LDA #02	256	85 FA	STA FA
281	85 D4	STA FLAG	258	60	RTN
282	20 60 02	JSR SUPER	252	85 FB	STA FB
283	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
284	A9 02	LDA #02	256	85 FA	STA FA
285	85 D4	STA FLAG	258	60	RTN
286	20 60 02	JSR SUPER	252	85 FB	STA FB
287	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
288	A9 02	LDA #02	256	85 FA	STA FA
289	85 D4	STA FLAG	258	60	RTN
290	20 60 02	JSR SUPER	252	85 FB	STA FB
291	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
292	A9 02	LDA #02	256	85 FA	STA FA
293	85 D4	STA FLAG	258	60	RTN
294	20 60 02	JSR SUPER	252	85 FB	STA FB
295	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
296	A9 02	LDA #02	256	85 FA	STA FA
297	85 D4	STA FLAG	258	60	RTN
298	20 60 02	JSR SUPER	252	85 FB	STA FB
299	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
300	A9 02	LDA #02	256	85 FA	STA FA
301	85 D4	STA FLAG	258	60	RTN
302	20 60 02	JSR SUPER	252	85 FB	STA FB
303	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
304	A9 02	LDA #02	256	85 FA	STA FA
305	85 D4	STA FLAG	258	60	RTN
306	20 60 02	JSR SUPER	252	85 FB	STA FB
307	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
308	A9 02	LDA #02	256	85 FA	STA FA
309	85 D4	STA FLAG	258	60	RTN
310	20 60 02	JSR SUPER	252	85 FB	STA FB
311	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
312	A9 02	LDA #02	256	85 FA	STA FA
313	85 D4	STA FLAG	258	60	RTN
314	20 60 02	JSR SUPER	252	85 FB	STA FB
315	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
316	A9 02	LDA #02	256	85 FA	STA FA
317	85 D4	STA FLAG	258	60	RTN
318	20 60 02	JSR SUPER	252	85 FB	STA FB
319	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
320	A9 02	LDA #02	256	85 FA	STA FA
321	85 D4	STA FLAG	258	60	RTN
322	20 60 02	JSR SUPER	252	85 FB	STA FB
323	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
324	A9 02	LDA #02	256	85 FA	STA FA
325	85 D4	STA FLAG	258	60	RTN
326	20 60 02	JSR SUPER	252	85 FB	STA FB
327	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
328	A9 02	LDA #02	256	85 FA	STA FA
329	85 D4	STA FLAG	258	60	RTN
330	20 60 02	JSR SUPER	252	85 FB	STA FB
331	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
332	A9 02	LDA #02	256	85 FA	STA FA
333	85 D4	STA FLAG	258	60	RTN
334	20 60 02	JSR SUPER	252	85 FB	STA FB
335	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
336	A9 02	LDA #02	256	85 FA	STA FA
337	85 D4	STA FLAG	258	60	RTN
338	20 60 02	JSR SUPER	252	85 FB	STA FB
339	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
340	A9 02	LDA #02	256	85 FA	STA FA
341	85 D4	STA FLAG	258	60	RTN
342	20 60 02	JSR SUPER	252	85 FB	STA FB
343	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
344	A9 02	LDA #02	256	85 FA	STA FA
345	85 D4	STA FLAG	258	60	RTN
346	20 60 02	JSR SUPER	252	85 FB	STA FB
347	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
348	A9 02	LDA #02	256	85 FA	STA FA
349	85 D4	STA FLAG	258	60	RTN
350	20 60 02	JSR SUPER	252	85 FB	STA FB
351	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
352	A9 02	LDA #02	256	85 FA	STA FA
353	85 D4	STA FLAG	258	60	RTN
354	20 60 02	JSR SUPER	252	85 FB	STA FB
355	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
356	A9 02	LDA #02	256	85 FA	STA FA
357	85 D4	STA FLAG	258	60	RTN
358	20 60 02	JSR SUPER	252	85 FB	STA FB
359	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
360	A9 02	LDA #02	256	85 FA	STA FA
361	85 D4	STA FLAG	258	60	RTN
362	20 60 02	JSR SUPER	252	85 FB	STA FB
363	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
364	A9 02	LDA #02	256	85 FA	STA FA
365	85 D4	STA FLAG	258	60	RTN
366	20 60 02	JSR SUPER	252	85 FB	STA FB
367	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
368	A9 02	LDA #02	256	85 FA	STA FA
369	85 D4	STA FLAG	258	60	RTN
370	20 60 02	JSR SUPER	252	85 FB	STA FB
371	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
372	A9 02	LDA #02	256	85 FA	STA FA
373	85 D4	STA FLAG	258	60	RTN
374	20 60 02	JSR SUPER	252	85 FB	STA FB
375	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
376	A9 02	LDA #02	256	85 FA	STA FA
377	85 D4	STA FLAG	258	60	RTN
378	20 60 02	JSR SUPER	252	85 FB	STA FB
379	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
380	A9 02	LDA #02	256	85 FA	STA FA
381	85 D4	STA FLAG	258	60	RTN
382	20 60 02	JSR SUPER	252	85 FB	STA FB
383	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
384	A9 02	LDA #02	256	85 FA	STA FA
385	85 D4	STA FLAG	258	60	RTN
386	20 60 02	JSR SUPER	252	85 FB	STA FB
387	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
388	A9 02	LDA #02	256	85 FA	STA FA
389	85 D4	STA FLAG	258	60	RTN
390	20 60 02	JSR SUPER	252	85 FB	STA FB
391	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
392	A9 02	LDA #02	256	85 FA	STA FA
393	85 D4	STA FLAG	258	60	RTN
394	20 60 02	JSR SUPER	252	85 FB	STA FB
395	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
396	A9 02	LDA #02	256	85 FA	STA FA
397	85 D4	STA FLAG	258	60	RTN
398	20 60 02	JSR SUPER	252	85 FB	STA FB
399	20 31 02	JSR TRANSF	254	A5 D3	LDA D3
400	A9 02	LDA #02	256	85 FA	STA FA

MOVE-A-BLOCK

Charles H. Parsons
80 Longview Rd.
Honore, 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 Butterfield's suggestion I felt the easiest way to do this would be to reprogramly enter P4 into the timer each time the interrupt (NMI) occurs. This theoretically produces a time of 249,856 microseconds or just under X second. The adjustment to Y 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 a single step operation. This is a price paid for giving the clock first priority. I also have a speaker hooked to P80 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 #21

0080	GSEC	2 Second Counter
0081	SEC	Second Counter
0082	MIN	Minute Counter
0083	HR	Hour Counter
0084	DAY	Day Counter For AM-PM
17FA		NMI Interrupt Pointers
17FB		
0360		Interrupt Routine
0361	PHA	Save A
0362	PHA	Save X
0363	PHA	Save Y
0364	PHA	Save Y
0365	LDA #383	Save Y
0366	STA TIME4	Save Y
0367	RIT TIMER7	Save Y
036A	RPL TM	Save Y
036D	INC QSEC	Save Y
036F	INC QSEC	Save Y
0371	LDA #304	Save Y
0373	CMP QSEC	Save Y
0375	BNE RTN	Save Y
0377	LDA #300	Save Y
0379	STA QSEC	Save Y
037B	CLC	Save Y
037C	SED	Save Y
037D	LDA SEC	Save Y
037F	ADC #301	Save Y
0381	STA SEC	Save Y
0383	CMP #360	Save Y
0385	BNE RTN	Save Y
0387	LDA #300	Save Y
0389	STA SEC	Save Y
038A	LDA MIN	Save Y
A582		Save Y

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

Display Clock On KIM-1 Readout

03C0	A900	LDA #300	Reset 2 Second Counter
03C2	8580	STA QSEC	Start Timer With Interrupt
03C4	A9F4	LDA #384	Start Here If Clock Is Running
03C6	8D0F17	STA TIMEF	Display Clock On KIM
03C9	A581	LDA SEC	
03CB	85P9	STA INH	
03CD	A582	LDA MIN	
03D1	A583	STA POINTL	
03D3	85FA	LDA HR	
03D5	201F1P	STA POINTH	
03D8	200003	JSR KIM	Escape To KIM
03DA	200002	JSR TIME	Minute Timer
03DB	202003	JSR BEEP	Sound On The Hour
03DE	209002	JSR UPDATE	Calendar
03E1	207502	JSR DSPDAY	Show Date
03E7			
03EA			
03ED			
03F0			
03F3			
03F6			
03F9			
4CC903			

PUT EA's (NOP) IN LOC. 03DB-03FB UNTIL
OTHER ROUTINES ARE ADDED. ~~FOR ADD-~~
TIONAL ROUTINES WILL BE IN AN UPCOMING
ISSUE - ECR

JMP DSP

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

038D	18	CLC	And Advance Minutes	Line	Code	Label	Instruction	Comment
038E	6901	ADC #301	Uhtll 60 Minutes	0300	206A1P	KIM	JSR GETKEY	Go Back To KIM If
0390	8582	STA MIN	Then Start Again	0303	C901		CMP #301	KIM Keyboard Is One
0392	C960	CMP #360	And Advance Hours	0305	D00D		BNE ENDR	Delay To Make Sure
0394	D019	RNE RTN		0307	201F1P		JSR SCANDS	
0396	A900	LDA #300		030A	206A1P		JSR GETKEY	
0398	8582	STA MIN		030D	C901		CMP #301	
039A	A583	LDA HR		030F	D003		BNE ENDR	
039C	18	CLC		4C051C	60	ENDR	JMP SAVE1	
039D	6901	ADC #301					RTN	
039F	8583	STA HR						
03A1	C912	CMP #312						
03A3	D002	BNE TH						
03A5	E684	INC DAY						
03A7	C913	CMP #313						
03A9	D004	BNE RTN						
03AB	A901	LDA #301						
03AD	8583	STA HR						
03AF	D8	CID						
03B0	A9P4	LDA #3P4						
03B2	8D0F17	STA TIMEF						
03B5	68	PLA						
03B6	88	TXA						
03B7	68	PLA						
03B9	AA	TXA						
03BA	40	RTI						

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 PART 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 MK508N. IT CAN BE DRIVEN DIRECT FROM ONE 8-BIT OUTPUT PART AND NEEDS AN INEXTENSIVE COLOR TV XTAL (3.58 MHz). THE MK508N IS AVAILABLE FOR \$8.95 FROM TRI-TEK, 6522 N. 43rd AVENUE, GLENDALE, ARIZONA 85301

6502 OP CODE TABLE

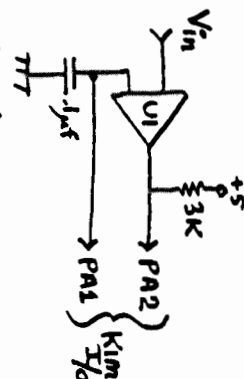
S/D ALLEN
507 Hill St. #5
Sunnyvale, CA 94085

0	1	2	4	5	6	8	9	A	C	D	E
BK ORA lmp 1,x	ORA lmp 1,x	ASL lmp 1,x	PPR lmp 1,x	ORA lmp 1,x	ASL lmp 1,x	PPR lmp 1,x	ORA lmp 1,x	ASL lmp 1,x	PPR lmp 1,x	ORA lmp 1,x	ASL lmp 1,x
BK ORA rel 1,y	ORA rel 1,y	ASL rel 1,y	PPR rel 1,y	ORA rel 1,y	ASL rel 1,y	PPR rel 1,y	ORA rel 1,y	ASL rel 1,y	PPR rel 1,y	ORA rel 1,y	ASL rel 1,y
JSH AND lmp 1,x	AND lmp 1,x	HOL lmp 1,x	PLP lmp 1,x	AND lmp 1,x	HOL lmp 1,x	PLP lmp 1,x	AND lmp 1,x	HOL lmp 1,x	PLP lmp 1,x	AND lmp 1,x	HOL lmp 1,x
BK AND rel 1,y	AND rel 1,y	HOL rel 1,y	PLP rel 1,y	AND rel 1,y	HOL rel 1,y	PLP rel 1,y	AND rel 1,y	HOL rel 1,y	PLP rel 1,y	AND rel 1,y	HOL rel 1,y
BK BOR lmp 1,x	BOR lmp 1,x	LSR lmp 1,x	PAA lmp 1,x	BOR lmp 1,x	LSR lmp 1,x	PAA lmp 1,x	BOR lmp 1,x	LSR lmp 1,x	PAA lmp 1,x	BOR lmp 1,x	LSR lmp 1,x
BK BOR rel 1,y	BOR rel 1,y	LSR rel 1,y	PAA rel 1,y	BOR rel 1,y	LSR rel 1,y	PAA rel 1,y	BOR rel 1,y	LSR rel 1,y	PAA rel 1,y	BOR rel 1,y	LSR rel 1,y
BK ADC lmp 1,x	ADC lmp 1,x	PLA lmp 1,x	ADC lmp 1,x	ADC lmp 1,x	PLA lmp 1,x	ADC lmp 1,x	ADC lmp 1,x	PLA lmp 1,x	ADC lmp 1,x	ADC lmp 1,x	PLA lmp 1,x
BK ADC rel 1,y	ADC rel 1,y	PLA rel 1,y	ADC rel 1,y	ADC rel 1,y	PLA rel 1,y	ADC rel 1,y	ADC rel 1,y	PLA rel 1,y	ADC rel 1,y	ADC rel 1,y	PLA rel 1,y
BK STA lmp 1,x	STA lmp 1,x	STX lmp 1,x	DEY lmp 1,x	STA lmp 1,x	STX lmp 1,x	DEY lmp 1,x	STA lmp 1,x	STX lmp 1,x	DEY lmp 1,x	STA lmp 1,x	STX lmp 1,x
BK STA rel 1,y	STA rel 1,y	STX rel 1,y	DEY rel 1,y	STA rel 1,y	STX rel 1,y	DEY rel 1,y	STA rel 1,y	STX rel 1,y	DEY rel 1,y	STA rel 1,y	STX rel 1,y
LDA LMA lmp 1,x	LMA lmp 1,x	LDA lmp 1,x	LMA lmp 1,x	LDA lmp 1,x	LMA lmp 1,x	LDA lmp 1,x	LMA lmp 1,x	LDA lmp 1,x	LMA lmp 1,x	LDA lmp 1,x	LMA lmp 1,x
LDA LMA rel 1,y	LMA rel 1,y	LDA rel 1,y	LMA rel 1,y	LDA rel 1,y	LMA rel 1,y	LDA rel 1,y	LMA rel 1,y	LDA rel 1,y	LMA rel 1,y	LDA rel 1,y	LMA rel 1,y
CPY CDP lmp 1,x	CDP lmp 1,x	CPY lmp 1,x	CDP lmp 1,x	CPY lmp 1,x	CDP lmp 1,x	CPY lmp 1,x	CDP lmp 1,x	CPY lmp 1,x	CDP lmp 1,x	CPY lmp 1,x	CDP lmp 1,x
CPY CDP rel 1,y	CDP rel 1,y	CPY rel 1,y	CDP rel 1,y	CPY rel 1,y	CDP rel 1,y	CPY rel 1,y	CDP rel 1,y	CPY rel 1,y	CDP rel 1,y	CPY rel 1,y	CDP rel 1,y
CPX SBC lmp 1,x	SBC lmp 1,x	CPX lmp 1,x	SBC lmp 1,x	CPX lmp 1,x	SBC lmp 1,x	CPX lmp 1,x	SBC lmp 1,x	CPX lmp 1,x	SBC lmp 1,x	CPX lmp 1,x	SBC lmp 1,x
CPX SBC rel 1,y	SBC rel 1,y	CPX rel 1,y	SBC rel 1,y	CPX rel 1,y	SBC rel 1,y	CPX rel 1,y	SBC rel 1,y	CPX rel 1,y	SBC rel 1,y	CPX rel 1,y	SBC rel 1,y
BEQ SBC lmp 1,y	SBC lmp 1,y	BEQ lmp 1,y	SBC lmp 1,y	SBC lmp 1,y	BEQ lmp 1,y	SBC lmp 1,y	SBC lmp 1,y	BEQ lmp 1,y	SBC lmp 1,y	SBC lmp 1,y	BEQ lmp 1,y
BEQ SBC rel 1,y	SBC rel 1,y	BEQ rel 1,y	SBC rel 1,y	SBC rel 1,y	BEQ rel 1,y	SBC rel 1,y	SBC rel 1,y	BEQ rel 1,y	SBC rel 1,y	SBC rel 1,y	BEQ rel 1,y

abs Absolute
abx absolute indexed using x register
acc accumulator
aby absolute indexed using y register
1,x indexed indirect using x register
1,y indexed indirect using y register
lmm immediate
lmp latched
lnd indirect
rel relative
sro zero page
spx zero page indexed using x register
syz zero page indexed using y register
spy zero page indexed using y register

least sig 4 bits

PAGE 9



$V_{in} = >.25v$ and $<4.0v$
 $U1 = LM311$

- IDEA FOR SOFTWARE DRIVER
- PROGRAM PAL AS OUTPUT, PA2 AS INPUT, WRITE "0" TO PA1
- LOAD TIMER WITH "FF"
- WRITE "1" TO PA1
- Loop 'til PA2 GOES HIGH.
- READ THE TIMER + SUBTRACT READING FROM "FF"
- WRITE "0" TO PA1 TO LET CAPACITOR DISCHARGE.
- Jump BACK TO STEP 2

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

```

0000 A9 27 START LDA #27          SBC value
0001 A2 3F GO                    Directional reg's
0002 42 07 LDA #7                set for input
0003 57 00 S7X PHDD              set for input
0004 BE 43 17 GO                 Directional reg's
0005 A2 3F GO                    SBC value
0006 A0 5E LDA #5E              High frequency
0007 2C 42 17 HIT SHD           Zero or one?
0008 0A 03 LDA #03              Low frequency
0009 A2 3F GO                    Directional reg's
0010 10 02 LDA #02              set for output
0011 49 80 BOR #80              Reverses output bit
0012 0B 42 17 S7A SHD           and send it
0013 83 44 17 S7Y CLKX1       Set timer
0014 2C 47 17 WAIT            ..and wait
0015 83 44 17 HIT CLKX0
0016 30 D9 BPL GO
0017 49 80 BOR #80
0018 0A 03 LDA #03
0019 A2 3F GO                    Directional reg's
0020 42 07 LDA #7                set for input
0021 57 00 S7X PHDD              set for input
0022 2C 47 17 WAIT            ..and wait
0023 83 44 17 HIT CLKX0
0024 30 D9 BPL GO
0025 10 FB BPL WAIT
0026 30 D9 BPL GO

```

Program TAPE DURE is fully relocatable.

SD value
Directional reg's
set for input
S7X PHDD
LDA #7
PDS (CONT) set
for input
LDA #7
S7X SHD
High frequency
Zero or one?
HIT SHD
LDA #03
Low frequency
Directional reg's
set for output
BOR #80
Reverses output bit
and send it
S7Y CLKX1
Set timer
..and wait

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

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 AUDIO IN and AUDIO OUT points. With the program running, start the recorder. All programs will be copied from one tape to the other.

Program TAPE DURE is fully relocatable.

up and sort a lot of tape a little at a time, etc.

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

NOTE: Load tape ID to 179 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 on 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 subr perhaps someone can debug that?

Caution the stack will be pushed down in page 1 quite far.

START -

```

A900 0052 BDEC17
0053 0055 4C7818 JMP #1878 return to KIM
0054 0058 85E6 STA E6 save
0055 005A ADEE17 LDA 17EE SAH
0056 005B 85EB STA EB
0057 005F ADEE17 LDA 17ED SAI
0058 0062 85EA STA EA
0059 0064 C5E7 CMP E7 end low
0060 0066 D007 BNE 07 end low
0061 0068 ADEE17 LDA 17EE SAH
0062 006B C5E8 CMP E8 end hi
0063 006D F005 BEQ 05 EXIT
0064 006F R005 LDA E6 get accum.
0065 0071 4CE900 JMP #00B9 jump to VEB.
0066 0074 4CXXXX JMP

```

LOAD TAPE BREAK -

Set up the following:

```

5800 17FE
00B6 XX reserved to save accumulator
00E7 ELEM tape, end low, end high, address
00E9 B0XXXX emulates "VEB"
00EC 4CEP17 JMP VEB + 3

```

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.

Keep up the good work.

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

Jim Butterfield, Toronto

TAPE DURE

ROBERT D. 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 TV.

THIS IS A MATH TEST

12
I 6

For the right answer 72 - YOUR 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 lines 266 for X & 265 for Y for
multiplication & at 365 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 266
110 IF I=2 GOTO 356
120 IF I=3 GOTO 566
130 IF I=4 GOTO 566
190 END
200 LET X=(RND (12)+1)
205 LET Y=(RND (12)+1)
210 IF X<16 GOTO 236
220 IF X>16 GOTO 246
230 PR " "
235 GOTO 266
240 PR " "
245 GOTO 266
260 IF Y<16 GOTO 286
270 IF Y>16 GOTO 296
280 PR " "
285 GOTO 366
290 PR " "
295 GOTO 366
300 PR " "
310 LET Q=X*Y
320 INPUT D
330 GOTO 126

```

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?

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

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

Interface Age; Nov 1976; p. 12-14; "Build a Stamp A to D", sample circuit,
6502 software

THE TITLE PUBLISHING
BIRMINGHAM, ALABAMA 35203

James R. Davis
ATTORNEY AT LAW

TELEPHONE
(919) 867-4444

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 avail-
able to me, and I have been primarily using stereo cassette tape decks manu-
factured by J. V. C. and Craig. When I first attempted to use Jim's pro-
gram I could get "fastape" and "speedtape" to run fine, but the "supertape" pro-
gram after initial synchronization and the reading of a few bytes, would be-
come unsynchronized, resulting in an abortive read. Also, the level settings
were extremely critical to even get initial synchronization. These observa-
tions were made by means of the use of a "VU Tape" program.

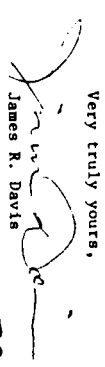
After some experimentation with the various values indicated on page 12
of Volume One, Issue Two of KIM-1 User Notes, I have found that loading hex
value 03 into address 01BE, and hex value 02 into address 01C0, seems
to give virtually fool-proof read/write performance to my system over an ex-
tremely wide range of input levels and types of cassette. I have used Maxwell
ID, 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
bus 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 unimpacted 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 pro-
gram 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

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

~ the editor ~

PAGE 10

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 opinion.

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 DYSR, and our task is this: If the accumulator is not less than DYSR, subtract DYSR and add one to QUOT, the quotient. We might be tempted to write:

```

CMP DYSR          ...elsewhere in the program:
  BCS SUB         SUB SEC
NEXT              SEC DYSR
                  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 DYSR
  BCC NEXT
  SEC DYSR
  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 #410          to test accumulator only
TAX              if positive, leave X at 10
BPL POS          ..else change X to 20
LDX #420
POS              ..coding continues
    
```

Are you starting to see the idea? Keep that flow in order whenever you can ... you'll end up with easier, shorter 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 #415 F 1	DOKEY CMP #410 numeric!
BNE MAL2	BCC MAL2
STA KDE	EOR #410 A becomes 0
RTS	STA KDE 0 or non-zero
CMP #410 A?	RTS
BNE MAL2	...continues
IDM #400	
STA KDE	
RTS	
MAL2 EDS 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
Niagara Falls, Ontario
715-2111
1-800-537

We are presently using the KIM-1 systems at the college to teach students in their third year operations, 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

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

maybe all
the educators
should get in
touch (?)

- 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, (JMP & JSR are used no much, I leave the B off.)
- Store A in port "A": 1700
Load A from 00E6
Rotate left data stored in 0300+X
LDXI EF A? EF
RORL 0300 3E 00 03
- A "paued" 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."
- 1.e., BITB IC53 2C 53 IC 2? 0 if bit 0 of accumulator is one
(01 stored in permanent memory IC53)
other single bit address/data in KIM 1: 1CCB/02, 1CAR/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/B CONVERTERS

WHAT HAVE YOU DONE
WITH YOUR KIM-1?

How 'bout some HARDWARE
STUFF?