Don Lancaster
SYNERGETICS

A TVT For Your KIM

... at a price you won't believe

Here is the great happening in TVT technology you have long been waiting for. Take a small, single-sided PC card with SIX integrated circuits on it, plug it into your KIM-1 or other microcomputer and display up to several thousand upper and lower case characters of your choice, all on an ordinary TV set with minimum modifications. Despite its "all the bells and whistles" performance, the cost of this new TVT approach is so ludicrously low that there's no comparing it with anything previously available. *If* you're a real dyed-in-the-wool scrounger (etch your own boards, steal sockets and connectors, burn your own PROMs, etc. . . .), you can put this together for around nine dollars plus the rapidly dropping cost of a character generator IC!

Complete kits and ready-to-go units are also available commercially, at somewhat higher but still unbelievable prices.

This new TVT development uses the microcomputer to do practically *all* of the system timing and control involved in a video display. This reduces the remaining interface circuitry to three hex inverters, two baby PROMs, and a character generator. Your computer alternates its compute and display modes, just as your KIM-1 now alternates between computer and keyboard modes. With fancy enough software, you can make this alteration nearly or completely transparent. More simply, you let the screen go blank when the computer is busy and doesn't want to talk to you anyway.

An entire book could be written on this whole new TVT ballgame. In fact . . . the book is called *Microprocessor Based Video Displays*, and Sams will print it. What we'll do here is lift just enough out of the book to show you how to build a video display for your KIM-1. The particular circuit is called a TVT-6L (L for lower case) and we'll show you how to build displays of 16 lines of 32 characters, 13 lines of 64 characters, and 25 lines of 64 characters, along with a fancy full-performance cursor that includes scrolling, erase to end of screen, full motions, and the usual goodies. The larger displays will take more memory than the bare bones KIM-1 has, so we'll show you one way to go with a KIM-1 and KIM-2 (4K add-on RAM) pair of cards.

How It Works

The block diagram of the TVT-6L is shown in Fig. 1a. An area of your microcomputer's regular memory is reserved for your display. On the minimum KIM-1, a 512 character, 16 x 32 display on pages 02 and 03 is a good starting point, although the TVT-6L card can work with any contiguous memory block from 0000 to 0FFF. Since the KIM uses parts of page 00 and 01 for its operating system, these usually aren't available for alphanumeric display use. For the larger displays with added RAM, memory locations from pages four through seven or else four through ten (0400-07FF or 0400-0AFF) are a good choice.

Besides these display memory pages, you'll need a place to put the SCAN program that tricks the KIM-1 and a TV set into talking to each other compatibly. Usually your SCAN program is around ninety words long. On the KIM-1, this is easily stuffed into the leftover scratchpad RAM starting at 1780.

Our DECODE read only memory is the heart of our TVT circuit. This PROM is activated by sending it an address from 2000 to EFFF. When activated, the DECODE memory causes a companion SCAN memory to force the microcomputer into a *scan* mode that advances the CPU's program counter 32, 64, or some other number of selected steps, *advancing once each microsecond*, binary counter style.

During this active horizontal scan time of usually 32 or 64 microseconds, all the memory in the microcomputer is sequentially interrogated on a *one memory slot per microsecond* basis. A new *upstream tap* is added to the memory to be displayed that *always* outputs data to the TVT-6L circuit, even and particularly when the display memory does NOT have access to the data bus.

So, during a scan mode, the display memory outputs characters to the TVT even though it does not have control of the data bus. The characters have the format shown in Fig. 1b, with an ASCII character using up the lower seven bits of the memory word. An optional cursor bit is placed on bit eight if wanted or needed. A zero in bit eight does nothing; a one optionally displays a winking cursor under both software and hardware control.

The lower seven bits of a character that were sent from the upstream tap go to a character generator IC1. IC1 also receives some "what row of dots do we want?" information from the DECODE read only memory IC2. This particular character generator has an internal video shift register to directly output serial video in a 7 x 11 dot matrix with descenders format. The internal video shift register in IC1 derives its load and clock timing pulses from the KIM system clock Ø2 by way of gated oscillator IC6.

The serial output video goes now to a new, simple, and super-important circuit called a *bandwidth enhancer* in IC5. The bandwidth enhancer *predistorts* the output video to exactly cancel the way your TV set is going to try and mess it up. With this circuit, it's a simple matter to output several thousand characters per frame and still stay within the ordinary video bandwidth of a plain old TV set. Our bandwidth enhancer simply makes the dots longer than the undots, with the amount of lengthening set by a CLARITY control that is tuned to your TV for the sharpest and brightest display.

Meanwhile, two other outputs from the DECODE memory IC2 go to a position delay circuit in IC4 to provide horizontal and vertical positioning. The delayed sync signals are combined with the enhanced video in *video combiner* IC5.

IC5 gives us *two* outputs. One is the usual monitor output with grounded sync tips and +2 volt white level, used with monitors and completely preconverted TV sets. Our second *TV* output is translated upwards to put the white level at +4 volts, the usual bias level needed to go *directly* into the base of the first video amplifier transistor in a portable, transformer operated, solid state, black and white TV. Thus, our *TV* output greatly simplifies direct video interface. As Fig. 7 will show us later on, all you do is rip off the headphone jack and use it as an automatic video changeover switch, defeat the sound trap, and that's all you need — at least for the 16 x 32 display.

Three switches on the TVT-6L let you program the module to suit your particular needs. One switch picks 32 versus 64 character lines when used with suitable scan software. A second switch gives you a choice of no cursor or of a winking underline cursor under software control. The final switch is the neatest one to watch since it gives you a choice of all upper case or mixed upper and lower alphanumerics.

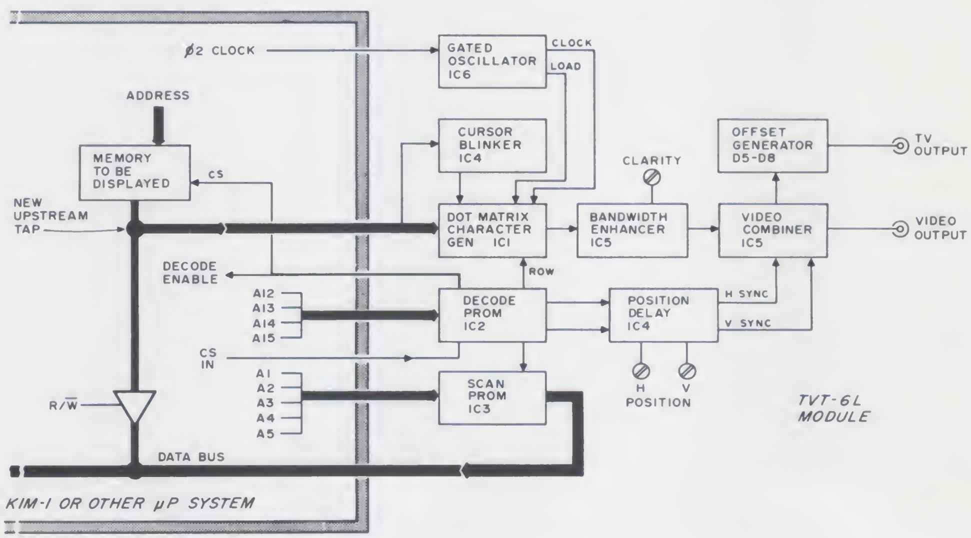


Fig. 1a. TVT-6L Block Diagram. The CPU does all the work.

Fig. 1b. Data bus definitions.

Scanning

A SCAN program activates the DECODE memory once each horizontal line, which

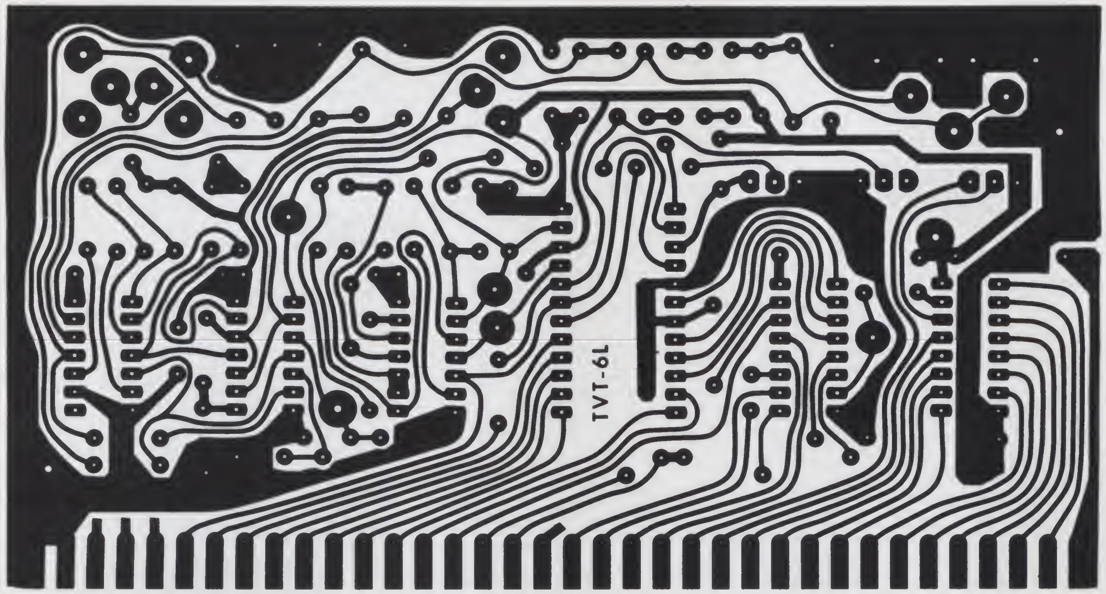


Fig. 4. TVT-6L printed circuit board layout (full size).

results in serial video being output for that particular line. If the program calls for a blank or retrace scan, all zeros are output, resulting in an all-black line. After a horizontal scan, the scan program computes the character and row information needed for the next line, and continues this way, on through one composite frame of fully interlaced video.

The design of a scan program is extra tricky since you have to control the exact number of microseconds everything takes to keep both the computer and the TV set happy. But once the scan program is designed and debugged, it's nothing but several dozen words of RAM or ROM available when needed to output the contents of the memory pages as video.

All our scan program does is cause the pages of memory reserved for characters to appear on the screen. *The SCAN program has absolutely nothing to do with how the characters get onto or off of that memory*, and couldn't care less. Your ordinary

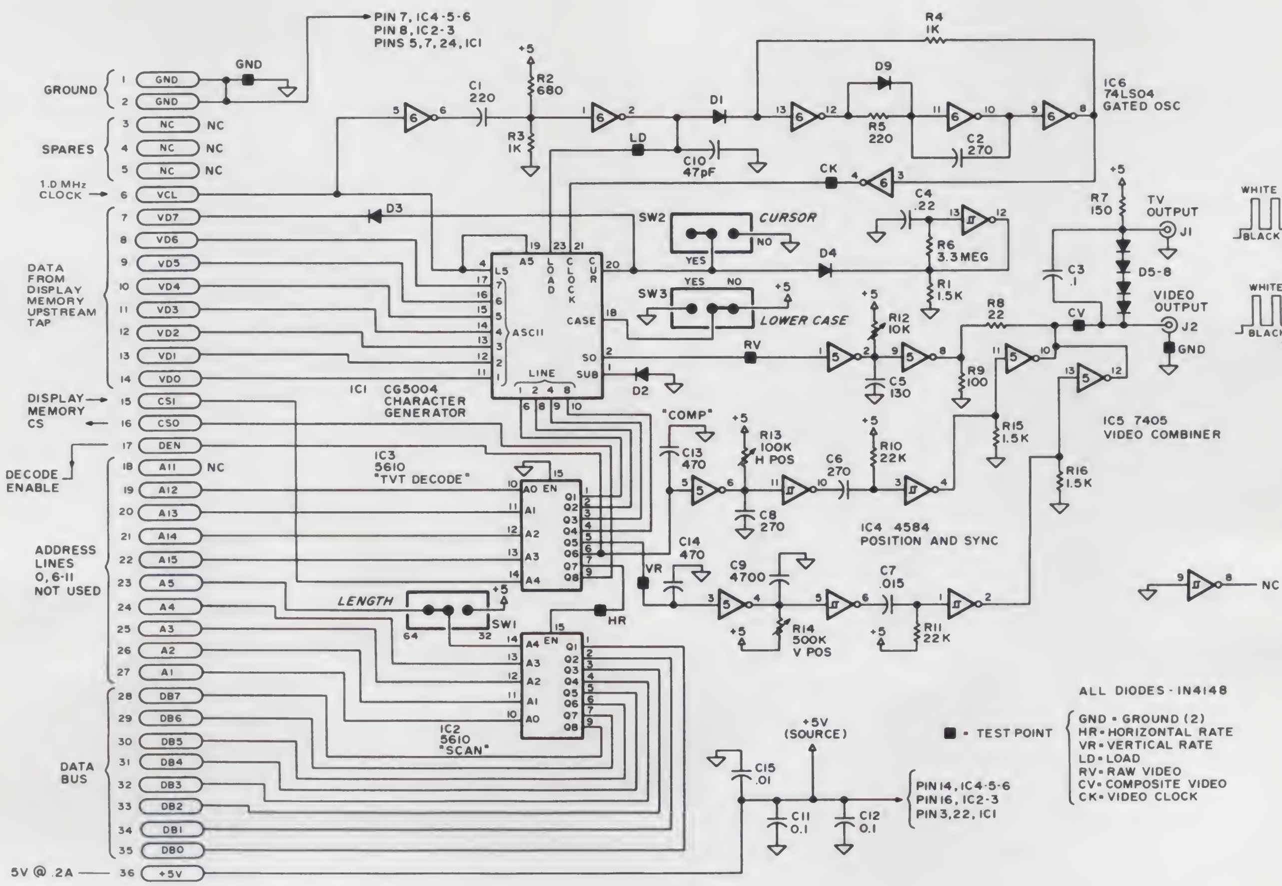


Fig. 2. TVT-6L Schematic.

WORD	NOTES	7	6	5	4	3	2	1	0
0	LDY 240_8	■	□	■	□	□	□	□	□
1	" "	■	□	■	□	□	□	□	□
2	" "	■	□	■	□	□	□	□	□
3	" "	■	□	■	□	□	□	□	□
4	" "	■	□	■	□	□	□	□	□
5	" "	■	□	■	□	□	□	□	□
6	" "	■	□	■	□	□	□	□	□
7	" "	■	□	■	□	□	□	□	□
8	" "	■	□	■	□	□	□	□	□
9	" "	■	□	■	□	□	□	□	□
10	" "	■	□	■	□	□	□	□	□
11	" "	■	□	■	□	□	□	□	□
12	" "	■	□	■	□	□	□	□	□
13	" "	■	□	■	□	□	□	□	□
14	" "	■	□	■	□	□	□	□	□
15	" "	■	□	■	□	□	□	□	□
16	" "	■	□	■	□	□	□	□	□
17	" "	■	□	■	□	□	□	□	□
18	" "	■	□	■	□	□	□	□	□
19	" "	■	□	■	□	□	□	□	□
20	" "	■	□	■	□	□	□	□	□
21	" "	■	□	■	□	□	□	□	□
22	" "	■	□	■	□	□	□	□	□
23	" "	■	□	■	□	□	□	□	□
24	" "	■	□	■	□	□	□	□	□
25	" "	■	□	■	□	□	□	□	□
26	" "	■	□	■	□	□	□	□	□
27	" "	■	□	■	□	□	□	□	□
28	" "	■	□	■	□	□	□	□	□
29	" "	■	□	■	□	□	□	□	□
30	" "	■	□	■	□	□	□	□	□
31	RTS 140_8	□	■	■	□	□	□	□	□

Fig. 3a. Truth Table for SCAN PROM, IC2.

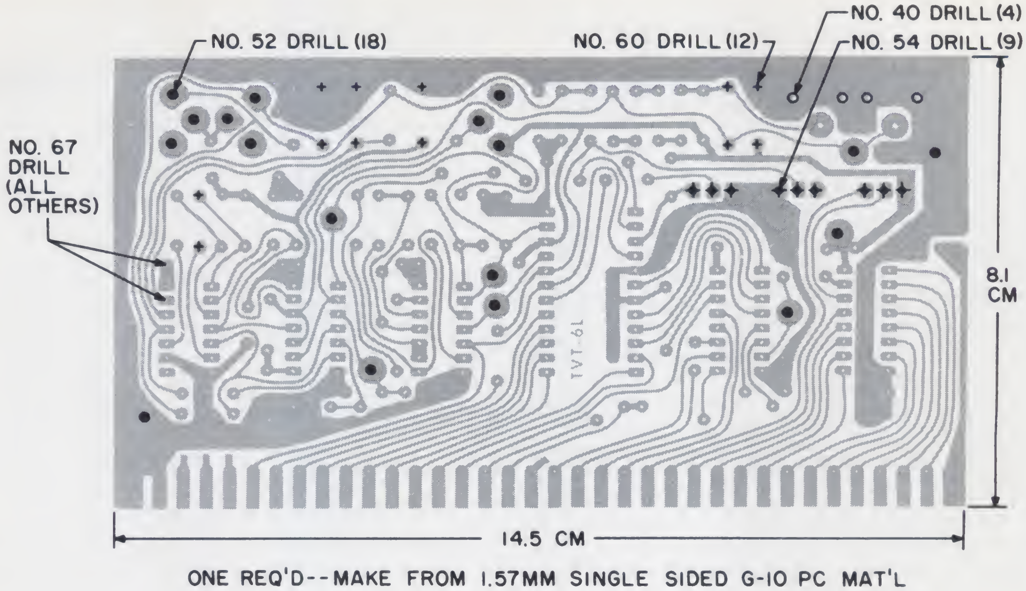


Fig. 5. Mechanical and drilling details for PC board.

WORD	NOTES	7	6	5	4	3	2	1	0
0	NORMAL 100	□	■	□	□	□	□	□	□
1	NORMAL 100	□	■	□	□	□	□	□	□
2	BLANK 040	□	□	■	□	□	□	□	□
3	LINE 1 041	□	□	■	□	□	□	□	■
4	LINE 2 042	□	□	■	□	□	□	■	□
5	LINE 3 043	□	□	■	□	□	□	■	■
6	LINE 4 044	□	□	■	□	□	■	□	□
7	LINE 5 045	□	□	■	□	□	■	□	■
8	LINE 6 046	□	□	■	□	□	■	■	□
9	LINE 7 047	□	□	■	□	□	■	■	■
10	LINE 8 050	□	□	■	□	■	□	□	□
11	LINE 9 051	□	□	■	□	■	□	□	■
12	LINE 10 052	□	□	■	□	■	□	■	□
13	LINE 11 053	□	□	■	□	■	□	■	■
14	V SYNC 120	□	■	□	■	□	□	□	□
15	NORMAL 100	□	■	□	□	□	□	□	□
16	NORMAL 300	■	■	□	□	□	□	□	□
17	NORMAL 300	■	■	□	□	□	□	□	□
18	BLANK 040	□	□	■	□	□	□	□	□
19	LINE 1 041	□	□	■	□	□	□	□	■
20	LINE 2 042	□	□	■	□	□	□	■	□
21	LINE 3 043	□	□	■	□	□	□	■	■
22	LINE 4 044	□	□	■	□	□	■	□	□
23	LINE 5 045	□	□	■	□	□	■	□	■
24	LINE 6 046	□	□	■	□	□	■	■	□
25	LINE 7 047	□	□	■	□	□	■	■	■
26	LINE 8 050	□	□	■	□	■	□	□	□
27	LINE 9 051	□	□	■	□	■	□	□	■
28	LINE 10 052	□	□	■	□	■	□	■	□
29	LINE 11 053	□	□	■	□	■	□	■	■
30	V SYNC 320	■	■	□	■	□	□	□	□
31	NORMAL 300	■	■	□	□	□	□	□	□

Fig. 3b. Truth Table for DECODE PROM IC3.

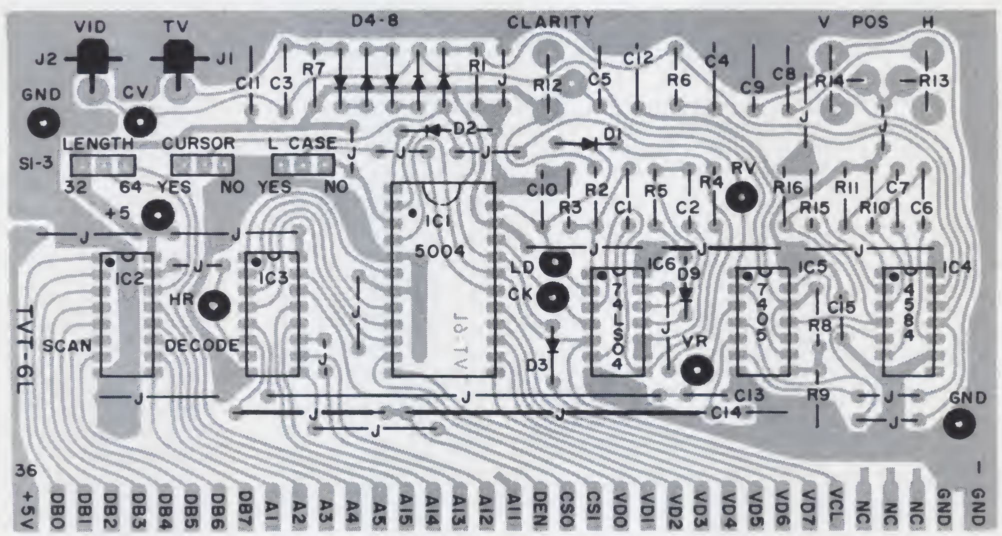


Fig. 6. Components placement overlay.

KIM-1 firmware can be used to load and dump memory to cassette tape. Your internal keypad can be used to put messages onto the screen by writing onto the memory pages. This trick gives you a zero cost ASCII keyboard and encoder, but at the hassle of having to write everything in hex rather than ASCII code.

For most uses, you'll want to add an external ASCII keyboard, entering on parallel A inputs and interrupting the Scan program every time you want to change a character. We'll be looking at a full scrolling cursor program later, but the important point now is that *you use whatever ordinary KIM-1 compatible programs you like to fill and empty the display memory pages*. Your separate Scan program simply puts the memory pages on the screen.

This way, you have total access to the screen memory at any time for any reason. Things like a displayed real time clock are trivial, and you can load and dump characters at a fantastic rate. With a simple Hex-to-ASCII adapter, you can also display op-code directly instead of alphanumeric characters. Note that this new TVT approach *isn't* DMA (Direct Memory Access) with its related drivers and access hassles. Your character memory is, looks, feels, and tastes just like any other memory in the microcomputer, since we've kept our upstream tap a secret from the CPU.

Building It

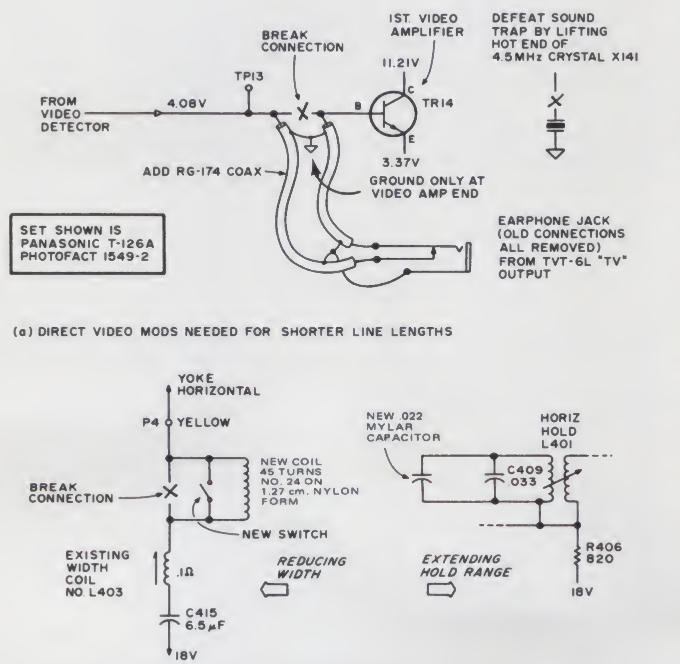
Fig. 2 shows you the schematic along with its parts list, while Fig. 3 gives you the truth tables for the DECODE and SCAN read only memories. Note that these are Tri-state 32 x 8 PROMs. Their programs obviously change if you use a system different from the KIM-1.

You'll find a full size printed circuit layout in Fig. 4, along with the mechanical and drilling details of Fig. 5. Components are located per the overlay of Fig. 6.

Start construction with the jumpers, using sleeving where shown. Follow this up with the nine test points and

C1	220 pF polystyrene
C2, 6, 8	270 pF polystyrene
C4	0.22 uF mylar
C3, 11, 12	0.1 uF mylar (Disks OK for C11, 12)
C5	130 pF polystyrene
C7	.015 uF mylar
C9	.0047 uF mylar
C10	47 pF polystyrene
C13, 14	470 pF disk
C15	.01 uF disk
D1-9	1N4148 or equivalent silicon computer diode. D1, D9 must be quality units.
IC1	CG5004L-1 character Generator IC (STD Microsystems)
IC2	IM5600 or equivalent 32 x 8 Bipolar Tri-state PROM "SCAN"
IC3	IM5600 or equivalent 32 x 8 Bipolar Tri-state PROM "DECODE"
IC4	4584 CMOS Hex Schmitt Trigger (Motorola)
IC5	7405 TTL Hex open collector inverter
IC6	74LS04 LS TTL hex inverter
J1, 2	PC Mount Phono Jack, Keystone 571
R1, 15, 16	1.5k, 1/4 watt carbon film resistor
R2	680 ohm, "
R3, 4	1k, "
R5	220 ohm, "
R6	3.3 Megohm, "
R7	150 ohm, "
R8	22 ohm, "
R9	100 ohm, "
R10, 11	22k, "
R12	10k upright trimmer potentiometer CTS U201 "CLARITY"
R13	100k, " "H POS"
R14	500k, " "V POS"
S1-3	SPDT miniature switch 3.17 mm pin centers
MISC:	PC Board, etched and drilled per Fig. 4; Test Point Terminals (9); PC Sockets, 24 pin (1), 16 pin (2), 14 pin (3): Matching connector (Amphenol 225 or equivalent); Sleeving; jumper material; solder.
NOTE:	The following are available from PAIA Electronics, Box 14539, Oklahoma City, OK, 37114:
	PC Board, etched and drilled, #TVT-6LB, $4.00
	Complete kit of all above parts #TVT-6LK, $59.95
	Assembled and Tested TVT-6L, #TVT-6LAT, $75.00
	KIM Coded Cassette, #TVT-6LC, $5.00

Complete Parts List, TVT-6L.



(a) DIRECT VIDEO MODS NEEDED FOR SHORTER LINE LENGTHS

(b) ADDITIONAL MODS THAT MAY BE NEEDED FOR 64 CHARACTER LINES

Fig. 7. Mods to TV set are greatly simplified thanks to already offset "TV" output on TVT-6L. DO NOT ATTEMPT DIRECT VIDEO ON A HOT CHASSIS (NO POWER TRANSFORMER) SET!

the low profile IC sockets, the switches, and the output phono jacks. Finally, add the resistors, pots, capacitors, and diodes. Be sure to note the polarity of each diode as it is added. Use fine solder and a small iron, and be sure to carefully double-check for any splashes or missed connections.

Software

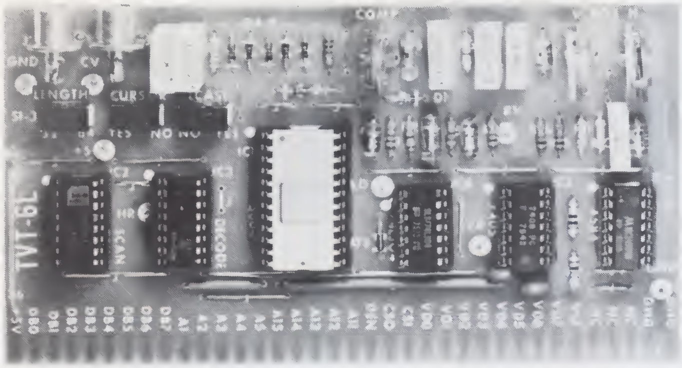
Table I shows us some tested and workable KIM software. Program 1-A is the scan program for a 16 x 32 fully interlaced 512 character display that can be moved around as needed. Program 1-B is a dual program that you can set up as 25 x 64 or 13 x 64 fully interlaced displays with larger characters. Finally, Program 1-C gives us a four-in-one full performance scrolling cursor. 1-C accepts an external ASCII keyboard on the parallel A inputs and works with any of the display formats by changing the key words as shown. This particular cursor system includes all the bells and whistles, such as full, rapid cursor motions in all directions, scrolling, erase to end of screen, and so on. It takes up most of page 01 in the KIM-1. You can easily make the cursor program longer for super fancy editing or shorter for a minimum sequential loading, per your choice.

Modifying your KIM

Table II gives you a complete listing of all pinouts on the TVT-6L along with the interface connections needed for either a KIM-1 or KIM-2 interface. The actual computer mods are outlined in Fig. 8. For the KIM-1, you add a connector along the top and make wiring pencil direct connections as shown. The foil is cut in ONE place along the 1K memory chip select line and a changeover switch is added. With the switch in the NORMAL position, the chip select is driven from K0 as usual. In the TVT position, the chip select line is driven from the TVT's CSO line, which is a *negative logic* OR of KO and the TVT's scan access.

For larger displays, you'll need extra memory. Fig. 8b shows us the modifications for a KIM-2 memory. These mods first convert the KIM-2 decodings so that the KIM-2 works on the second, third, fourth, and fifth "K" of memory, or addresses 0400-13FF. Only addresses 0400 through 0FFF *may be* used for TVT page storage, although the remaining space is available for other computer use. The rest of the KIM-2 mods are similar to those on the KIM-1.

Even if you are going to use extra memory (who isn't?), convert the KIM-1 anyway as it is the simplest and best way to get started with your video displays. The changeover switch lets you run with the TVT-6L out of



Closeup of TVT-6L module. Three of the six! ICs used are hex inverters. Switches give choice of line length, upper and lower case, cursor. Twin jacks give either monitor video or already-translated TV video. Both outputs are enhanced for minimum bandwidth needs. Module is adaptable to many popular microprocessors and microcomputers.

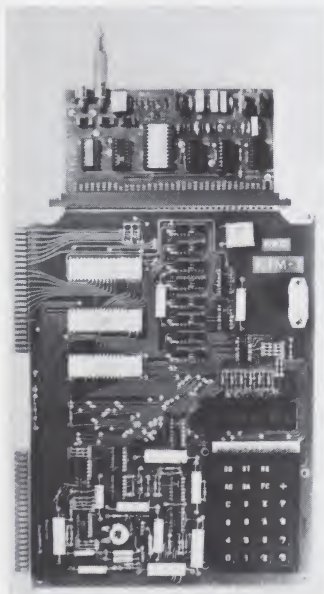
the socket. In one position, the computer works normally; in the other, it will work as a TVT or transparently so long as the TVT is in the socket and so long as memory locations 2000 to EFFF are not called.

TV Mods

The TV output with its +4 white bias level greatly simplifies your direct video interface. Fig. 7 shows how to "borrow" the headphone jack on a Panasonic T126A and convert it into an automatic video changeover switch. The only new parts needed are two short pieces of miniature coax. The sound trap is defeated by lifting the hot end of the 4.5 Megahertz crystal.

This type of conversion works on any small screen, solid state TRANSFORMER OPERATED portable B and W television set, so long as the set needs a bias voltage around +4 volts at the input to the first video stage.

In order to use the 64 character displays on an ordinary TV without extensive video bandwidth changes, the television's horizontal frequency is run much lower than normal, around 11 kHz. This means that you'll most likely need a width and hold modification for 64 or other long character lines. On the set shown, you can use a coil of 48 turns of #24 wire on a 1.27 cm diameter nylon form in series with the existing



The KIM-1 with a difference. Cable at top delivers enhanced video direct to your TV set or monitor.

width coil. A new hold mylar capacitor of one third the normal value, or .022 uF, is added in parallel to C409 to extend the hold range downwards.

Note that the reduced horizontal frequency and reduced width are only needed on 64 character lines. The shorter 32 character lines run at normal horizontal speed. This tradeoff buys us a lot in the way of being able to scan characters with the CPU in the first place and eliminates any need for video bandwidth extension, so it is well worth the simple and reversible mods needed. Clip-on RF modulators can also be used as shown in the *TV Typewriter Cookbook* and *Microprocessor Based Video Displays*, again thanks to the reduced horizontal rate with long line lengths.

With any TV modification, be sure to have a SAMS photofact on hand and get expert help if you've never done a video input conversion before. NEVER ATTEMPT DIRECT VIDEO INTO A HOT CHASSIS OR TRANSFORMERLESS TELEVISION SET.

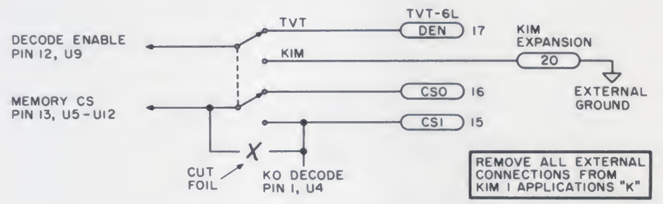
Initial Checkout

Always have a good oscilloscope on hand for your initial checkout, and always do your first check on a KIM-1 in the 16 x 32 utility scan program 1A mode. Don't worry about doing anything initially except displaying code that already happens

1. Add a new 36 pin, single readout connector along the top of the KIM-1 above the crystal. Small "L" brackets can be added to use existing holes.
2. Make short and direct wire connections as shown in Table II. Use a wiring pencil for all connections *except* +5 and GND, which should be short lengths of #18 wire.

Do not use ribbon cable or attempt extending the TVT-6L.

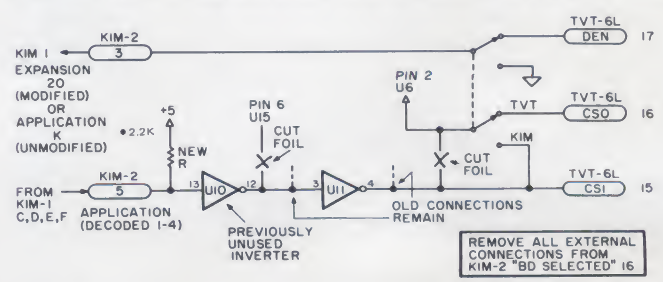
3. Break ONE foil run as shown, and add a DPDT changeover switch:



With the switch in the TVT position, operation is totally transparent so long as the TVT-6L is in its socket and addresses 2000-EFFF aren't called.

Fig. 8a. Modifying your KIM-1 for the TVT-6L.

1. Add a new 36 pin, single readout connector along the left edge of the card, the side away from the regulator. Small "L" brackets can use existing holes if one of the handle eyelets are replaced with a #6 screw.
2. Make short and direct wire connections as shown in Table II. Use a wiring pencil for all connections *except* +5 and GND.
3. Break TWO foil leads as shown, and add a DPDT changeover switch:



Note that we now have a new input pin on Connector 5 that is driven by KIM-1 decodings K1, 2, 3, and 4 in parallel from Application connector C, D, E, and F.

We also have a new output pin on Connector 3 that provides a ground for the KIM-1 Decode Enable. This is connected to Application Connector K on an *unmodified* KIM-1 and to Expansion Connector 20 or a KIM-1 modified per Fig. 8a.

Note further that BD SELECTED output Connector 16 is not used.

These modifications cause your KIM-2 to respond to addresses 0400-13FF. The program address switches are no longer used.

Fig. 8b. Modifying your KIM-2 for the TVT-6L.

to be on pages 02 and 03.

Center the three controls, switch to "32", cursor OFF and Lower Case OFF, plug the TVT-6L into your KIM-1, insert IC2 and IC3 *only*, and apply power. Go through the usual initialization, putting the KIM-1 in its binary mode with the interrupt returning you to keyboard operation. Then, try to operate the KIM-1 with a simple program on page zero to make sure the chip select and decode enable logic on the TVT-6L is passing things through properly and transparently.

Now, check address 2000 with your keyboard monitor. It should contain an A0. *If it doesn't, stop immediately and find out why*! Check the next 29 locations for more A0s followed by two 60s followed by another string of 30 A0s and so on. You should now be able to write and single step a simple program that will transfer control from KIM to TVT back to KIM again (see Example 1).

Your KIM-1 should start at 0000, jump to 2000, index sixteen times by twos to 201E, return to 0003 and stick there in the trap. *Don't go on till the KIM and TVT can pass control back and forth to each other.*

Next, add IC6 and check testpoints LD and CK with a decent scope. The waveforms should look exactly like Fig. 9. In particular, they should be clean and stable. The clock should have eight narrow positive clock pulses between load commands. *Do not omit checking these waveforms.*

Add the rest of the ICs,

0000	JSR	20	00	20	Go to TVT
0003	JMP	4C	03	00	Trap

Example 1.

Table I. Some software.

A. *Program for a 16 line, 32 character per line Interlaced TVT-6L Raster Scan:*

uP — 6502 | Start — JMP 17A6 | Displayed 0200-03FF
System — KIM-1 | End — Interrupt | Program Space 1780-17d4

Upper Address (178A0								Lower Address (1789)							
*	*	*	*	0	0	1	V8	V4	V2	V1	H16	H8	H4	H2	H1

0, 1, F — normal program (no tvt)
2 — blank scan
3 — scan row 1
4 — scan row 2
. (etc) ......
d — scan row 11
E — vertical sync pulse

Tape Ident — 6A
Program Length — 85 words + 1 word page zero (EC)

1780	CLC	18			Clear Carry
1781	STA	8d	(8A)	(17)	Store Upper Address
1784	PHA	48			Equalize 10 microseconds
1785	PLA	68			continued
1786	BNE	d0	00		continued
1788	JSR	20	00	20	///Character Scans 0-11///
178b	ADC	69	10		Increment Character Scan Counter
178d	CMP	C9	E0		Character Scan Counter Overflow?
178F	BCC	90	F0*		No, Scan next row of character
1791	TAX	AA			Save Upper Address
1792	LDA	Ad	(89)	(17)	Get Lower Address
1795	ADC	69	1F		Increment Lower Address; Save carry
1797	STA	8d	(89)	(17)	Restore Lower Address; Save carry
179A	TXA	8A			Get Upper Address
179b	ADC	69	40		Reset Upper Address; add carry
179d	BNE	d0	00		Equalize 3 microseconds
179F	JSR	20	04	20	///Blank Character Scan 12///
17A2	CMP	C9	24		Is it the "17th" row of characters?
17A4	BCC	90	dA*		No, start a new row of characters
START 17A6	LDA	A5	(EC)		Get Interlace Word
17A8	ADC	69	7F		Change Field via Carry bit
17AA	BCS	B0	05*		Jump if Even Field
17AC	STA	8d	(EC)	E0	Odd Field V Sync; Restore Interlace
17Af	LDX	A2	36		Load Odd (short) # of blank scans
17b1	LDY	A0	05		Equalize 31 microseconds
17b3	DEY	88			continued
17b4	BPL	10	Fd*		continued
17b6	BCC	90	05*		Jump if odd field
17b8	STA	8d	(EC)	E0	Even Field V Sync; Restore Interlace
17bb	LDX	A2	37		load Even (long) # of blank scans
17bd	JSR	20	1E	20	///1st V Blanking scan///
17C0	PHA	48			Equalize 9 Microseconds
17C1	PLA	68			continued
17C2	CLD	d8			continued
17C3	LDA	A9	00		Initialize Lower Address
17C5	STA	8d	(89)	(17)	continued
17C8	LDA	A9	22		Initialize Upper Address
17CA	STA	8d	(8A)	(17)	continued
17Cd	JSR	20	00	20	///Rest of V Blanking scans///
17d0	DEX	CA			One less scan
17d1	BMI	30	Ad*		Start Character Scan
17d3	BPL	10	Ed*		Repeat V Blanking Scan

NOTES: TVT-6L must be connected and both the SCAN and DECODE PROMs must be in circuit for program to run.

Both 17AC and 17b8 require that page 00 be enabled when page E0 is addressed. This is done automatically in the KIM-1 decode circuitry.

Location 00EC on page zero is reserved as an interlace storage bit.

Step 1788 goes to where the upper address stored in 178A and the lower address stored in 1789 tells it to. Values in these slots continuously change throughout the program.

For a 525 line system, use 17b0 34 and 17bC 35 and a KIM-1 crystal of 992.250 kHz. This is ONLY needed for a video superposition or tilting applications; the stock 1 MHz

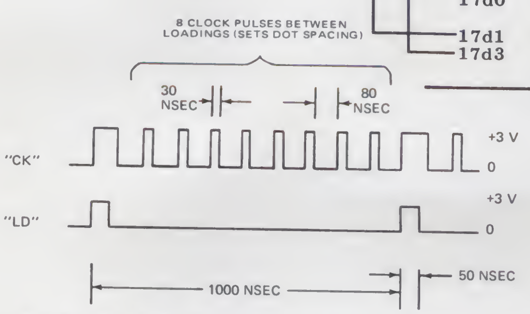


Fig. 9. Key high frequency waveforms. Clock "CK" must be clean, jitter free, and have narrow positive duty cycle shown.

crystal is used for ALL OTHER uses.

Normal program horizontal frequency is 15,873.015 kHz; Vertical 60.0114. 63 microseconds per line, 264.5 lines per field; 2 fields per frame 529 lines total.

TVT-6L switch must be in the "32" position.

( ) Denotes an absolute address that is program location sensitive.

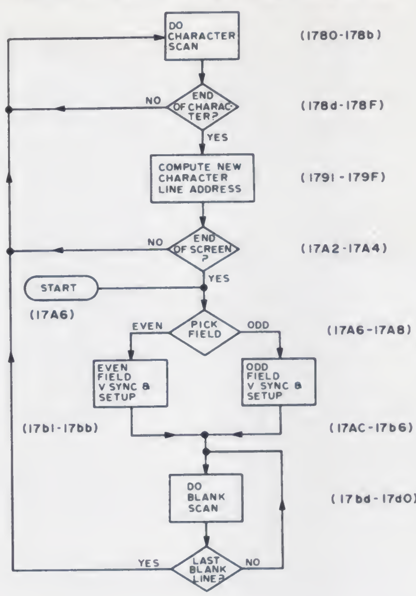
* Denotes a relative branch that is program length sensitive.

TO DISPLAY OTHER PAGES, USE:

PAGES DISPLAYED	17A3	17C9	TVT CONNECTION
0000-01FF	22	20	KIM-1
0200-03FF	24	22	KIM-1
0400-05FF	26	24	KIM-2
0600-07FF	28	26	KIM-2
0800-09FF	2A	28	KIM-2
0A00-0BFF	2C	2A	KIM-2
0C00-0DFF	2E	2C	KIM-2
0E00-0FFF	30	2E	KIM-2

FOR HIGHER PAGES, MOVE CONTENTS TO 0200-03FF or 0400-05FF

FLOWCHART 16X32 INTERLACED SCAN TVT-6L NO.6A



B. *Program for a 13 line or a 25 line, 64 character per line interlaced TVT-6L Raster Scan:*

uP — 6502
System — KIM-1, 2

Start — JMP 17bF
End — Interrupt

Displayed 04C0-0AFF
Program Space 1780-17E2

Upper Address 1788

*	*	*	*	PH	PL	V8	V4

Lower Address 1787

V2	V1	H32	H16	H8	H4	H2	H1

- 0.1, F — normal program (no tvt)
- 2 — blank scan
- 3 — scan row 1
- 4 — scan row 2
- . . . etc . . . .
- d — scan row 11
- E — vertical sync pulse

Tape Ident — 6B
Program Length 99 words +
1 word page zero (EC)

1780	LDA	A9	(24)		Initialize Upper Address
1782	STA	8d	(88)	(17)	Store Upper Address
1785	NOP	EA			Equalize 2
1786	JSR	20	00	20	///Character Scans 0-11////
1789	ADC	69	20		Increment Character Gen by 2
178b	CMP	C9	E0		Is it scan 12 or 13?
178d	BCC	90	F3*		No, Do next character scan
178F	PHA	48			Save Upper Address
1790	LDA	AD	(87)	(17)	Get Lower Address
1793	ADC	69	3F		Increment L; Set C on V2 Overflow
1795	STA	8d	(87)	(17)	Restore L; save carry
1798	PLA	68			Get Upper Word
1799	NOP	EA			Equalize 2
179A	JSR	20	0C	20	///Blank scans 12, 13///
179d	ADC	69	40		Add Carry; Reset Upper Address
179F	CMP	C9	2b		Was this the last line of characters?
17A1	BCC	90	dF*		No, Scan a new line of characters
17A3	LDA	A5	(EC)		Get Interlace Word
17A5	ADC	69	7F		Set Carry if Odd Field Finished
17A7	BCC	90	0F*		Start Even Field if Carry Set
17A9	STA	8d	(EC)	E0	Even V Sync + Replace Interlace
17AC	LDX	A2	0E		Load Even #VB Scans -2
17AE	LDA	A9	24		Initialize Even Upper Address
17b0	STA	8d	(81)	(17)	continued
17b3	LDA	A9	2b		Initialize Even Character End Compare
17b5	STA	8d	(A0)	(17)	continued
17b8	LDY	A0	07		Equalize 41 microseconds
17bA	DEY	88			continued
17bb	BPL	10	Fd*		continued
17bd	BCS	b0	0F*		Skip if Even Field
17bF	STA	8d	(EC)	E0	Odd V Sync + Replace Interlace
17C2	LDX	A2	0F		Load Odd #VB Scans -2
17C4	LDA	A9	34		Initialize Odd Upper Address
17C6	STA	8d	(81)	(17)	continued

START → 17bF

```
17C9   LDA   A9   3b            Initialize Odd Character End Compare
17Cb   STA   8d   (A0)   (17)          continued
17CE   JSR   20   3F     20     ///1st V Blanking Scan/////
17d1   LDA   A9   C0            Initialize Lower Address
17d3   STA   8d   (87)   (17)          continued
17d6   BMI   30   00            Equalize 3 microseconds
17d8   CLD   d8                 Equalize 4 microseconds
17d9   NOP   EA                        continued
17dA   JSR   20   00     20     ///Rest of V Blanking Scans///
17dd   DEX   CA                 One Less Scan
17dE   BMI   30   A0*           Start Character Scan
17E0   CLC   18                 Clear Carry
17E1   BPL   10   F5*           Repeat V Blanking Scan
```

NOTES:

TVT-6L must be connected and both the SCAN and DECODE PROMS must be in circuit for program to run.

Both 17A9 and 17bF require that page 00 be enabled when page E0 is addressed. This is done automatically in the KIM-1 decode circuitry.

Location 00EC on page zero is reserved as an interlace storage bit.

Step 1786 goes to where the upper address stored in 1788 and the lower address stored in 1787 tells it to. Values in these slots continuously change throughout the program.

Values in slots 1781 (Upper address start) and 17A0 (Character end compare) alternate with the field being scanned.

Horizontal Scan Frequency = 11.494 kHz. Vertical frequency = 60.0222 Hertz. 87 microseconds per line 191.5 lines per field; 2 fields per frame, 383 lines total.

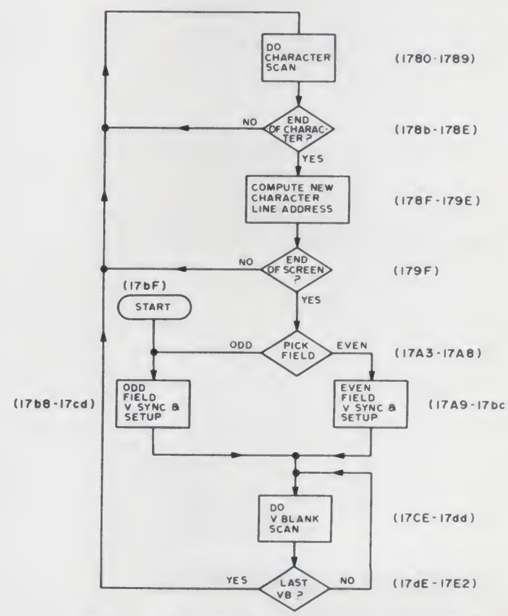
TVT-6L switch must be in the "64" position.

( ) Denotes an absolute address that is program location sensitive.

* Denotes a relative branch that is program length sensitive.

Program may be used for 13 x 64 large characters or 25 x 64 small characters by changing the following slots:

	13 x 64	25 x 64
178A	10	20
17Ad	14	0E
17Af	24	24
17b4	28	2b
17C3	15	0F
17C5	24	34
17CA	28	3b



FLOWCHART 13 X 64 OR 25 X 64 INTERLACED SCAN NO 6B

C. *Program for a Four-in-One full performance Scrolling Cursor:*

uP — 6502 Start — $\overline{\text{IRQ}}$ Program Space 0100-01dF
System — Kim 1,2 End — RTI + Two words page zero (ED,EF)

Input to parallel Word A

0	A7	A6	A5	A4	A3	A2	A1

IRQ
10 usec.

Clear — CAN (18)
Carriage Return — CR (0d)
Cursor Up — VT (0b)
Cursor Down — LF (0A)
Cursor Left — BS (08)

Cursor Right — HT (09)
Cursor Home — SOH (0A)
Scroll Up — DC1 (11)
Erase To End —ETX (03)
Enter -- all characters and all *unused* CTRL commands

```
Enter via
IRQ
       |
       +->0100   PHA   48                  Save A
          0101   LDY   A0   00             Reset Y Index
          0103   LDA   A5   (EE)           Get Cursor and test for range
          0105   CMP   C9   04             Is cursor below maximum?

[0147]<-0107     BCS   b0   3E*            No, Home Cursor
        0109     CMP   C9   02             Is cursor above minimum?
[0147]<-010b     BCC   90   3A*            No, Home cursor
        010d     LDA   b1   (Ed)           Get Old Cursed character

        010F     AND   29   7F             Erase Old Cursor
        0111     STA   91   (Ed)           Replace character without cursor
        0113     LDA   Ad   00     17      Get New character from A parallel Int.
        0116     CMP   C9   20             Is it a character to be entered?

[013E]<-0118     BCS   b0   24*            Yes, go and enter character
        011A     CMP   C9   18             Clear Screen?
[015E]<-011C     BEQ   F0   30*            Yes, go clear screen
        011E     CMP   C9   0d             Return carriage?

[0152]<-0120     BEQ   F0   30*            Yes, go return carriage
        0122     CMP   C9   0b             Move cursor up?
[0194]<-0124     BEQ   F0   6E*            Yes, move cursor up
        0126     CMP   C9   0A             Move Cursor down?

[0166]<-0128     BEQ   F0   3C*            Yes, move cursor down
        012A     CMP   C9   09             Move cursor right?
[0158]<-012C     BEQ   F0   2A*            Yes, move cursor right
        012E     CMP   C9   08             Move Cursor left?

[01A7]<-0130     BEQ   F0   75*            Yes, Move cursor to left
        0132     CMP   C9   01             Home Cursor?
[0147]<-0134     BEQ   F0   11*            Yes, Home cursor
        0136     CMP   C9   11             Scroll Up?

[0175]<-0138     BEQ   F0   3b*            Yes, Scroll Up
        013A     CMP   C9   03             Erase to End of Screen?
[01b1]<-013C     BEQ   F0   73*            Yes, Erase to End of Screen
        013E     CLD   d8                  Assure Hex arithmetic mode

        013F     JSR   20   (d3)   (01)    ////Enter Character via Sub////
        0142     BNE   d0   06*            Did Screen Overflow?
        0144     JMP   4C   (75)   (01)    Select Scroll or Wraparound
        0147     JSR   20   (C2)   (01)    ////Home cursor via sub////

        014A     LDA   b1   (Ed)           ////Restore Cursor////
        014C     ORA   09   80             Add Cursor to cursed character
        014E     STA   91   (Ed)           Restore cursed character
        0150     PLA   68                  Restore Accumulator

[OUT]<--0151     RTI   40                  Return to Scan
        0152     LDA   A5   (Ed)           ////Carriage Return///(get cursor)
        0154     ORA   09   1F             Move cursor all the way right
        0156     STA   85   (Ed)           Restore Cursor

        0158     JSR   20   (d5)   (01)    Increment cursor
[0142]<-015b     JMP   4C   (42)   (01)    Scroll or wraparound if needed; finish
        015E     JSR   20   (C2)   (01)    ////Clear///// (home cursor)
        0161     JSR   20   (Cb)   (01)    clear screen via subroutine

[0147]<-0164     BEQ   F0   E1*            Finish
        0166     LDA   A5   (Ed)           ////Cursor Down////(get cursor)
        0168     CLC   18                  Clear Carry
        0169     ADC   69   20             Move cursor down one line

        016b     STA   85   (Ed)           Restore Cursor
      +-016d     BCC   90   03*            Overflow of page?
      | 016F     JSR   20   (d9)   (01)    Yes, increment next higher page
[0142]<=0172     JMP   4C   (42)   (01)    Scroll or wraparound if needed; finish

        0175     JSR   20   (C2)   (01)    /////Scroll Up////(home cursor)
      +-0178     LDY   A0   20             Add offset to index
      | 017A     LDA   b1   (Ed)           Get offset indexed character
      | 017C     LDY   A0   00             Remove offset from index
      ^
      | 017E     JSR   20   (d3)   (01)    Enter moved character and increment
      +-0181     BNE   d0   F5*            Repeat?
        0183     CLC   18                  Clear Carry
        0184     LDA   A9   03             Set A to page of last line

        0186     STA   85   (EE)           Set Cursor to page of last line
        0188     LDA   A9   E0             Load A to start of last line
        018A     STA   85   (Ed)           Set Cursor to start of last line
[014A]<-018C     BCS   b0   bC*            Finish if carry set

        018E     JSR   20   (Cb)   (01)    Clear last line
        0191     SEC   38                  Set Carry
[0184]<-0192     BCS   b0   F0*            Restore cursor to start of last line
        0194     LDA   A5   (Ed)           /////Cursor Up///(get cursor)

        0196     SEC   38                  Set Carry
        0197     SBC   E9   20             Move Up one line
        0199     STA   85   (Ed)           Restore Cursor
[014A]<-019b     BCS   b0   Ad*            Underflow of page?

        019d     DEC   C6   (EE)           Yes, Decrement page
        019F     LDA   A9   01             Set A to page below home page
        01A1     CMP   C5   (EE)           Did screen underflow?
[014A]<-01A3     BNE   d0   A5*            No, Finish
```

load program 1A, switch to single step off, jump to 17A6 and hit GO. Your first check should be that the program *will* run, returning to the keyboard monitor when you hit stop and picking up on go again. Addresses should always be within the program bounds of being somewhere between 2000 and EFFF or somewhere between 1780 and 17d3.

Check test point VR for a one microsecond pulse every 16.7 milliseconds. If your scope has trouble with low duty cycle waveforms, you can try pin 6 of IC4, which should be a one millisecond or so pulse every 16.7 milliseconds. For the acid test, switch to *line* sync. This pulse is your vertical sync pulse. It should wander around *very* slowly with respect to the power line sync. This pulse is created both by the hardware and your SCAN program. *Stop right here till you have it there and stable.*

Now, plug in your fully modified TV or monitor to the VID output, or else a Fig. 8 modified TV to the TV output. You should have a random but stable display of characters, along with some weird control symbols. Position them and sharpen them with the controls. The CLARITY control makes the characters brighter in one direction and sharper in the other – pick what you like. At this point you should have a stable and attractive display. Use *minimum* contrast for sharpest characters.

The rest should be downhill all the way. Check the LCASE switch and the CURSOR switch. Around half the characters should wink cursors at you, since the cursor recognizes any bit eight set as a cursor and since you have a random page load, rather than a page of characters with a single cursor location.

If everything checks out so far, you can now go on to longer character lines, external keyboards, cursor loading (don't forget to load the IRQ

```
[0147]<-01A5   BEQ   F0   A0*          Yes, Home cursor
        01A7   DEC   C6   (Ed)         ///Cursor Left///(decrement cursor)
        01A9   LDA   A9   FF           Set A to page underflow
        01Ab   CMP   C5   (Ed)         Test for page underflow

[019d]<-01Ad   BEQ   F0   (EE)         Change page if off page
[014A]<-01AF   BNE   d0   99*          Finish if on page
        01b1   LDA   A5   (EE)         /////Erase to EOS///(get cursor)
        01b3   PHA   48                Save Upper Cursor location on stack

        01b4   LDA   A5   (Ed)         Get Lower Cursor location
        01b6   PHA   48                Save Lower Cursor location on stack
        01b7   JSR   20   (Cb)   (01)  Clear to End of Screen
        01bA   PLA   68                Get lower cursor location off stack

        01bb   STA   85   (Ed)         Restore lower cursor
        01bd   PLA   68                Get upper cursor location off stack
        01bE   STA   85   (EE)         Restore upper cursor
[014A]<-01C0   BNE   d0   88*          Finish

        01C2   LDA   A9   00           ///Subroutine-HOME CURSOR////
        01C4   STA   85   (Ed)         Set lower cursor to home value
        01C6   LDA   A9   02           Load A with home page value
        01C8   STA   85   (EE)         Set upper cursor to home page

        01CA   RTS   60                Return to main cursor program
      ┌─01Cb   LDA   A9   20           ///Subroutine-ENTER SPACES////
      ▲ 01Cd   JSR   20   (d3)   (01)  Enter space via character entry sub
      └─01d0   BNE   d0   F9*          Repeat if not to end of screen

        01d2   RTS   60                Return to main cursor program
        01d3   STA   91   (Ed)         ///Subroutine-ENTER AND INCREMENT///
        01d5   INC   E6   (Ed)         Enter character and increment
      ┌─01d7   BNE   d0   06*          Overflow of page?

      │ 01d9   INC   E6   (EE)         Yes, Increment cursor page
      ▼ 01db   LDA   A9   04           Load A with page above display
      │ 01dd   CMP   C5   (EE)         Test for Overflow
      └─01dF   RTS   60                Return to main cursor program
```

vector!), and so on, but don't try too much at once. *Always* get the utility 512 character basic display up on a KIM-1 before trying anything fancy. Thanks to the total software control, once you *are* up and working and confident, there's practically no limit to how fancy you get with your display.

More details on all this will appear in *Microprocessor Based Video Displays*, along with such options as a Hex-ASCII converter that displays *Super Front Panel* Op-Code (your whole program at once — how's that for a debug aide?); color graphics options, use of different character generators, different microprocessors, and so on. Watch for it. ■

NOTES: For auto-scrolling use 0145 75. For wraparound, use 0145 47.

$\overline{IRQ}$ vector must be stored in 17FE 00 and 17FF 01.

Total available stack length is 32 words. Approximately 16 are used by operating system, cursor, and scan program. Stack must be initialized to 01FF as is done in KIM operating system. For 30 additional stack locations, relocate subroutines starting at 01C2 elsewhere. For total stack availability, relocate entire program elsewhere.

To protect page, load 00F1 04. To enable entry, load 00F1 00

Cursor address is stored at 00Ed low and 00EE high on page zero.

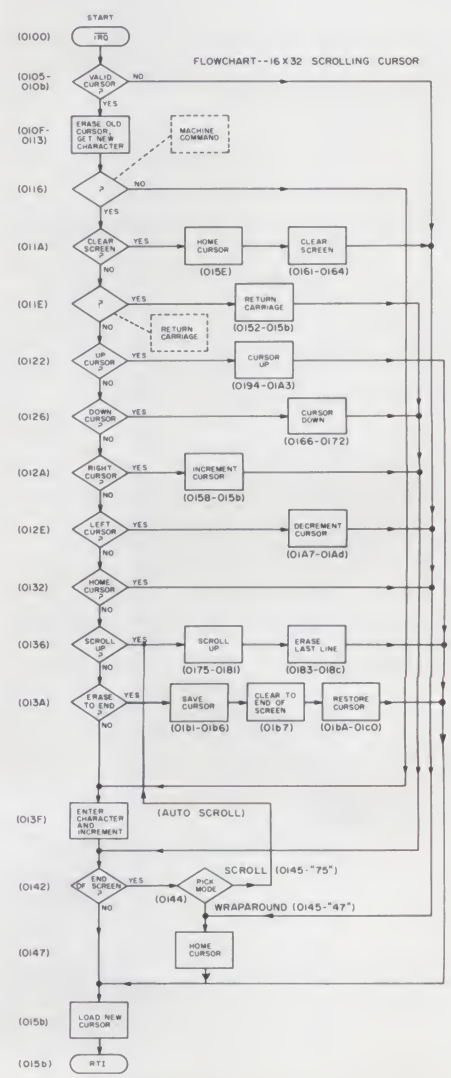
To display cursor, load 014d 80. To not display cursor, load 014d 00

* Denotes a relative branch that is program length sensitive

( ) Denotes an absolute address that is program location sensitive

To match this program to the scan program, change the following slots:

	16 x 32 KIM1 0200-03FF	16 x 32 KIM2 0400-05FF	13 x 64 KIM2 04C0-07FF	25 x 64 KIM2 04C0-0AFF
0106	04	06	08	0b
010A	02	04	04	04
0155	1F	1F	3F	3F
016A	20	20	40	40
0179	20	20	40	40
0185	03	05	07	0A
0189	E0	E0	C0	C0
0198	20	20	40	40
01A0	01	03	03	03
01C3	00	00	C0	C0
01C7	02	04	04	04
01dC	04	06	08	0b



Pin	Ident	Function	Load	8-A KIM-1 Connections	8-B KIM-2 Connections
1, 2*	GND	Ground Return — Heavy foil or wire	—	Expansion 22	Connector 1
3, 4, 5	NC	No Connection — reserved	—	—	—
6*	VCL	Video Clock Ø2	1 LSTTL	Expansion U	Pin 4 of U10
7	VD7	Cursor from Display memory	1 LSTTL	Pin 12 of U5	Pin 2 of U3
8	VD6	ASCII Bit 7 from Display memory	1 NMOS	Pin 12 of U6	Pin 6 of U3
9	VD5	ASCII Bit 6 from Display memory	1 NMOS	Pin 12 of U7	Pin 10 of U2
10	VD4	ASCII Bit 5 from Display memory	1 NMOS	Pin 12 of U8	Pin 2 of U2
11	VD3	ASCII Bit 4 from Display memory	1 NMOS	Pin 12 of U9	Pin 6 of U2
12	VD2	ASCII Bit 3 from Display memory	1 NMOS	Pin 12 of U10	Pin 10 of U1
13	VD1	ASCII Bit 2 from Display memory	1 NMOS	Pin 12 of U11	Pin 2 of U1
14	VD0	ASCII Bit 1 from Display memory	1 NMOS	Pin 12 of U12	Pin 6 of U1
15*	CSI	Chip Select *from* Enable Decoding	1 LSTTL	Pin 1 of U4	Pin 4 of U11
16*	CSO	Chip Select *to* Display Memory	TTL Out	Pin 13 of U5-U12	Pin 2 of U6
17*	DEN	Decode Enable *to* KIM	TTL Out	Pin 12 of U4	Connector 3
18	A11	No Connection — reserved	—	—	—
19	A12	Address Line 12	1 LSTTL	Expansion P	Connector R
20	A13	Address Line 13	1 LSTTL	Expansion R	Connector S
21	A14	Address Line 14	1 LSTTL	Expansion S	Connector T
22	AB15	Address Line 15	1 LSTTL	Expansion T	Connector U
23	A5	Address Line 5	1 LSTTL	Expansion F	Connector H
24	A4	Address Line 4	1 LSTTL	Expansion E	Connector F
25	A3	Address Line 3	1 LSTTL	Expansion D	Connector E
26	A2	Address Line 2	1 LSTTL	Expansion C	Connector D
27*	A1	Address Line 1	1 LSTTL	Expansion B	Connector C
28	DB7	Data Bus 7	TTL TS Out	Expansion 8	Connector 8
29	DB6	Data Bus 6	TTL TS Out	Expansion 9	Connector 9
30	DB5	Data Bus 5	TTL TS Out	Expansion 10	Connector 10
31	DB4	Data Bus 4	TTL TS Out	Expansion 11	Connector 11
32	DB3	Data Bus 3	TTL TS Out	Expansion 12	Connector 12
33	DB2	Data Bus 2	TTL TS Out	Expansion 13	Connector 13
34	DB1	Data Bus 1	TTL TS Out	Expansion 14	Connector 14
35	DB0	Data Bus 0	TTL TS Out	Expansion 15	Connector 15
36*	+5V	+5 Volt Supply	200 ma	Expansion 21	Connector Y

NOTES: (See * Above)

Pins 1, 2 — Ground should be heavy foil or #18 wire — all other connections are wire pencil short leads. *Do not use ribbon cables or attempt extension.*

Pin 6 — Video clock must load character generator only when data output is stable and valid. Clock Ø2 on the KIM.

Pins 15, 16 — Chip select line from decoding to display memory is broken by cutting the foil and then replaced with a negative logic OR (positive AND) of the original chip select and the TVT chip select. *See Figure 8b.*

Pin 17 — Decode Enable output goes low when TVT is *not* scanning; goes high otherwise. Decoding must be disabled during active scans to allow SCAN memory access to data buss. *See Figure 8b.*

Pin 27 — Address line A0 is not used in the TVT module as the SCAN memory indexes every *second* microsecond. A0 is used in the display memory addressing.

Pin 36 — +5 power borrowed from computer. Extra noise on the +5 line will cause skewed or akward characters; may be fixed usually with extra bypassing. Use heavy foil or #18 wire.

Table II. TVT-6L Interface.