Don Lancaster
Synergetics

TVT Hardware Design

. . . Part 1: instruction decoder and scan

Don Lancaster will soon have a new book released by Howard W. Sams called The Cheap Video Cookbook. *The following is the first of a two-part series, taken from the book, and is an elaboration on the hardware design that went into the TVT-6L. The TVT-6L is an entirely new concept in low-cost video displays that was originally presented in the June 1977 issue of* Kilobaud. *— John.*

An interface card has to hold most of the dedicated circuitry needed between a microprocessor and a TV set. You'll find a block diagram of a typical card shown in Fig. 1. Depending on its design details, you can use this type of card with graphics, alphanumerics or a combination of the two.

The *instruction decoder* is the central controller of a microprocessor-based video display. It's usually a small bipolar PROM. When activated by the scan program, the instruction decoder decides when a scan of video is needed and what video is going to be produced. Control signals are delivered to the rest of the interface circuitry by the instruction decoder. These signals include sync pulses and disabling signals that go back to make sure nothing else tries to use the microcomputer at the same instant that the TVT needs it.

The *scan microprogram* generator is a second PROM that outputs a scan microinstruction to the microprocessor. This PROM is activated by the instruction decoder every time a scan of video is wanted.

The *data-to-video converter* is usually a dot-matrix character-generator integrated circuit for alphanumeric use, and is usually a shift register or a shift-register and data-selector combination for graphics use. This converter block converts code stored in the computer's display memory and received by way of the new upstream tap into serial video you can display. (The upstream tap is a point in the memory between the memory ICs and the data bus drivers, i.e., "raw" memory output.) The instruction decoder controls the data-to-video converter by telling it which row of dots to output on a character or which part of a word to output for a graphics format.

High Frequency Timing controls the serial-video dot output and rate. This block can be a hex inverter gated oscillator driven from the microcomputer's main clock. A *cursor* circuit may also crop up in alphanumeric TVTs. The cursor introduces the winking underline or box that shows us the next character location. The cursor circuit usually is made up of a low-frequency oscillator and some gating.

The *sync and position* block takes horizontal and vertical timing signals from the instruction decoder. It then delays these timing signals as needed for positioning. After this, it goes on to shape these sync commands into the proper time widths for TV use.

Our *video output circuitry* combines video and sync and then provides a composite output suitable for monitor, rf modulator or direct television video interface. One important part of the output circuitry is the *bandwidth enhancer*. This simple compensation circuit is usually

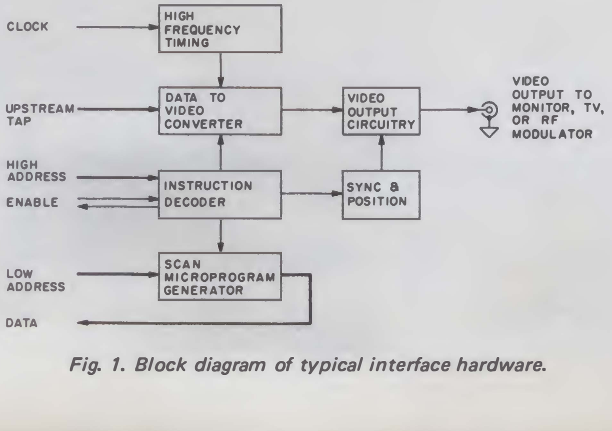


Fig. 1. Block diagram of typical interface hardware.

The

INSTRUCTION
DECODER

must:

- * Activate the scan microprogram when a scan is needed.
- * Disable everything else trying to use the computer CPU when a scan is needed.
- * Select the right graphics format or alphanumeric dot row.
- * Output sync pulses as needed.
- * Otherwise not interfere with normal computer operation.

Fig. 2. The Instruction Decoder PROM is the key controlling block of the interface hardware.

included to predistort the output video in anticipation of how the TV set will try to mess it up. The result is denser, sharper characters for a given TV's bandwidth, and is one of the keys to displaying long character lines on an ordinary TV set.

Let's take a closer look at these interface hardware blocks and see just what is involved in their design and use.

Instruction Decoder

Our instruction decoder PROM is the control center for TVT use of a microprocessor. The important functions of the instruction decoder are summarized in Fig. 2. It has to tell the microprocessor when to generate a scan of characters or graphics chunks, and it has to pick the right part of whatever you are going to display. Furthermore it has to firmly take over command of the microcomputer when the TVT is in use. When not in use, the instruction decoder has to make the interface hardware appear invisible to normal computer operation.

A 256-bit bipolar PROM of 32 words of eight bits each is a good choice as an instruction decoder. This is the smallest PROM you can buy, costing under $2. Important advantages of using a PROM for the instruction decoder are the flexibility of assigning what each address does, the ease of changes, and the single-IC simplicity of board layout.

Fig. 3 shows one good way to use a 32 x 8 PROM as an instruction decoder. We input high-order address lines A15, A14, A13 and, optionally, A12. We use A12 when we need 16 total instructions. We can omit A12 when eight or fewer instructions will do the job.

There are eight output leads available. One of these is used for a *decode enable* that takes over command from the computer's normal address decoding. On a KIM this is line KO, and is low for normal computer use and high for TVT use. A second output is a chip select command that goes low when we want to activate the display memory as far as the upstream tap. A third output drives our SCAN microprogram generator, going low to produce a scan microinstruction. Two sync outputs are needed, both horizontal and vertical. Often the *decode enable* output can double as a horizontal sync output, saving us a pin.

The remaining four output lines can be used to format the output data. In alphanumerics, these can be the

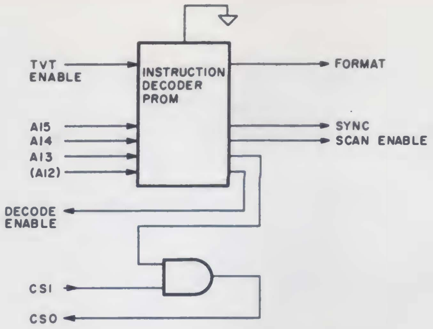


Fig. 3. Instruction Decoder PROM using external gate for display memory chip select. This allows other, non-TVT, uses of high-order address lines.

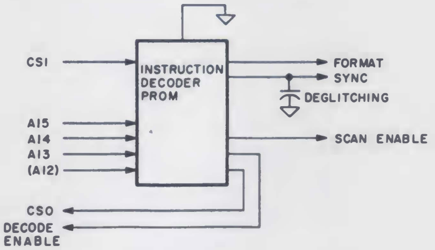


Fig. 4. Instruction Decoder PROM, with internal gating for display memory chip select. This saves a gate but produces output glitches and reserves most high-order addresses for exclusive TVT uses.

	INPUTS			OUTPUTS							
				Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
	WORD #	WHAT DOES THIS WORD DO ?	HEX OP-CODE	CS OUT	SCAN ENABLE	DECODE ENABLE	VERT SYNC	CG LINE "8"	CG LINE "4"	CG LINE "2"	CG LINE "1"
NORMAL CHIP SELECT	0	NORMAL	4Ø	□	■	□	□	□	□	□	□
	1	NORMAL	4Ø	□	■	□	□	□	□	□	□
	2	BLANK SCAN	2Ø	□	□	■	□	□	□	□	□
	3	LINE 1 SCAN	21	□	□	■	□	□	□	□	■
	4	LINE 2 SCAN	22	□	□	■	□	□	□	■	□
	5	LINE 3 SCAN	23	□	□	■	□	□	□	■	■
	6	LINE 4 SCAN	24	□	□	■	□	□	■	□	□
	7	LINE 5 SCAN	25	□	□	■	□	□	■	□	■
	8	LINE 6 SCAN	26	□	□	■	□	□	■	■	□
	9	LINE 7 SCAN	27	□	□	■	□	□	■	■	■
	10	LINE 8 SCAN	28	□	□	■	□	■	□	□	□
	11	LINE 9 SCAN	29	□	□	■	□	■	□	□	■
	12	LINE 10 SCAN	2A	□	□	■	□	■	□	■	□
	13	LINE 11 SCAN	2b	□	□	■	□	■	□	■	■
	14	VERTICAL SYNC	5Ø	□	■	□	■	□	□	□	□
	15	NORMAL	4Ø	□	■	□	□	□	□	□	□
TVT CHIP SELECT	16	NORMAL	CØ	■	■	□	□	□	□	□	□
	17	NORMAL	CØ	■	■	□	□	□	□	□	□
	18	BLANK SCAN	2Ø	□	□	■	□	□	□	□	□
	19	LINE 1 SCAN	21	□	□	■	□	□	□	□	■
	20	LINE 2 SCAN	22	□	□	■	□	□	□	■	□
	21	LINE 3 SCAN	23	□	□	■	□	□	□	■	■
	22	LINE 4 SCAN	24	□	□	■	□	□	■	□	□
	23	LINE 5 SCAN	25	□	□	■	□	□	■	□	■
	24	LINE 6 SCAN	26	□	□	■	□	□	■	■	□
	25	LINE 7 SCAN	27	□	□	■	□	□	■	■	■
	26	LINE 8 SCAN	28	□	□	■	□	■	□	□	□
	27	LINE 9 SCAN	29	□	□	■	□	■	□	□	■
	28	LINE 10 SCAN	2A	□	□	■	□	■	□	■	□
	29	LINE 11 SCAN	2B	□	□	■	□	■	□	■	■
	30	VERTICAL SYNC	dØ	■	■	□	■	□	□	□	□
	31	NORMAL	CØ	■	■	□	□	□	□	□	□

6L-D12
PROM NUMBER

□ = "0"
■ = "1"
(POSITIVE LOGIC)

USE FOR TUT-6L
12 DOT LINE
ALPHANUMERIC
SCANS.

Fig. 5a. Truth table for Alphanumeric Decode PROM 6L-D12.

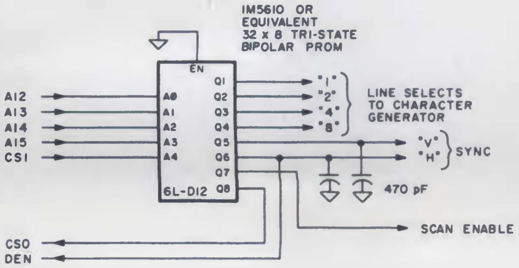


Fig. 5b. Connections for Alphanumeric Decode PROM 6L-D12.

INPUTS				OUTPUTS							
				Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
	WORD #	WHAT DOES THIS WORD DO ?	HEX OP-CODE	CS OUT	SCAN ENABLE	DECODE ENABLE	V SYNC	LOWER SELECT	BLANK	SPARE	SPARE
TVT ENABLED	0	NORMAL	4Ø	□	■	□	□	□	□	□	□
	1	•	4Ø	□	■	□	□	□	□	□	□
	2	•	4Ø	□	■	□	□	□	□	□	□
	3	BLANK SCAN	2Ø	□	□	■	□	□	□	□	□
	4	LOWER CHUNK SCAN	2C	□	□	■	□	■	■	□	□
	5	UPPER CHUNK SCAN	24	□	□	■	□	□	■	□	□
	6	VERTICAL SYNC	5Ø	□	■	□	■	□	□	□	□
	7	NORMAL	2Ø	□	■	□	□	□	□	□	□
	8	•	CØ	■	■	□	□	□	□	□	□
	9	•	CØ	■	■	□	□	□	□	□	□
	10	•	CØ	■	■	□	□	□	□	□	□
	11	BLANK SCAN	AØ	■	□	■	□	□	□	□	□
	12	LOWER CHUNK SCAN	2C	□	□	■	□	■	■	□	□
	13	UPPER CHUNK SCAN	24	□	□	■	□	□	■	□	□
	14	VERTICAL SYNC	dØ	■	■	□	■	□	□	□	□
	15	NORMAL	CØ	■	■	□	□	□	□	□	□
TVT DISABLED	16	•	4Ø	□	■	□	□	□	□	□	□
	17	•	4Ø	□	■	□	□	□	□	□	□
	18	•	4Ø	□	■	□	□	□	□	□	□
	19	•	4Ø	□	■	□	□	□	□	□	□
	20	•	4Ø	□	■	□	□	□	□	□	□
	21	•	4Ø	□	■	□	□	□	□	□	□
	22	•	4Ø	□	■	□	□	□	□	□	□
	23	•	4Ø	□	■	□	□	□	□	□	□
	24	•	CØ	■	■	□	□	□	□	□	□
	25	•	CØ	■	■	□	□	□	□	□	□
	26	•	CØ	■	■	□	□	□	□	□	□
	27	•	CØ	■	■	□	□	□	□	□	□
	28	•	CØ	■	■	□	□	□	□	□	□
	29	•	CØ	■	■	□	□	□	□	□	□
	30	•	CØ	■	■	□	□	□	□	□	□
	31	•	CØ	■	■	□	□	□	□	□	□

7-G PROM NUMBER

□ = "0"
■ = "1"
(POSITIVE LOGIC)

USE FOR TVT-7 GRAPHICS SCANS.

Fig. 6a. Truth table for Graphics Decode PROM 7-G.

The

SCAN
MICROPROGRAM
GENERATOR

must:

- * Generate the right coding to sequentially scan a row of video.
- * Optionally provide for alphanumeric memory repacking.
- * Be transparent during other computer uses.

Fig. 7. The Scan PROM generates the long microinstruction needed to sequentially output a row of characters or graphics dots.

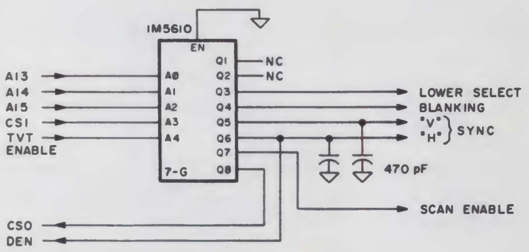


Fig. 6b. Connections for Graphics Decode PROM 7-G.

three or four "what line is it?" row commands that go to the character generator. For graphics, we can use a blanking output and an upper/lower output.

Our 32 x 8 PROM has a fifth input. We can pick just what we are going to do with it. In Fig. 3 we've used an external AND gate to combine the normal computer chip select with the TVT chip select to provide a composite CSO that activates the display memory as needed. This external gate is physically an AND gate and is shown with its usual positive logic symbol, but in reality it is used as an "either input low gives a low output," or as its DeMorgan equivalent OR gate.

We can call our fifth PROM input a *TVT enable* and activate it from some external logic. This lets you use the higher-order memory slots for other things besides TVT use. Letting the TVT enable low will let the TVT work; when it is high, the TVT remains inactive and the computer is free to do whatever else it wants with its high address lines. For all-the-time TVT use, simply ground this input.

Note that we have to keep our instruction decoder outputs active at all times to prevent messing up the decode enable commands. This usually means that the PROM's own enable input must stay grounded at all times. Thus, we must switch our PROM outputs from an active to a passive state as far as TVT operation is concerned; but we must never actually float the Tri-state outputs.

We also have the option of using our fifth PROM address input as a display memory chip select input *from* the computer. This internalizes the AND gate used for the chip selects as shown in Fig. 4. We did this on the TVT-6L (*Kilobaud* No. 6, June 77) as part of the mania for doing an entire video display in only six integrated circuits. There are two penalties to pay when you use internal display memory CS selection. One is that your outputs glitch, which means you have to crudely filter the sync outputs. The second is that you can't use many of the higher address locations for anything but TVT use.

Fig. 5a is the truth table for an alphanumeric instruction decoder having an internal chip select. This is the PROM used on the TVT-6L. Locations 0-15 are selected if the decoder is to pass through an existing low CSI. Locations 16-31 are selected if the TVT is only to provide its own CSO when needed to the display memory. The only output difference you'll see between these two halves of the truth table is the CSO output itself.

The remaining inputs are driven from A12 through A15 and select normal computer operation, a blank scan, a vertical sync pulse or scan of lines one through eleven. Outputs include the four character-generator line commands, the vertical sync output, the scan enable for the microprogram generator, the decode enable for the computer and the chip select output for the display memory. Typical connections are shown in Fig. 5b.

Your turn: Show a truth table for a similar alphanumeric PROM whose fifth input is a TVT enable line. Show how external logic can switch between TVT operation and other use of high-order address slots.

A graphics decode truth table that is used on the TVT-7 is shown in Fig. 6a, along with its typical connections in Fig. 6b. Only eight input address decodings are necessary, so A12 is no longer needed. This frees two PROM inputs; one is used as a CSI chip select input and the second as a TVT enable line.

Scan Microprogram Generator

The *scan microprogram generator* is a second PROM used as part of our interface hardware. Its purpose is to force a SCAN instruction onto the microprocessor. The microprocessor, in turn, gives us a sequential one-character-per-microsecond code output that lasts for as many characters or chunks as you want in a horizontal line. Fig. 7 sums up what our scan microprogram generator has to do.

To produce a scan, a scan software program calls a subroutine at an address that activates the instruction decoder. The instruction decoder then activates the scan microprogram generator, which produces the microcode needed for a sequential scan. For the 6502, coding

INPUTS				OUTPUTS							
				Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
	WORD #	WHAT DOES THIS WORD DO ?	HEX OP-CODE	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DBØ
NOT USED FOR 32 CHARACTER LINES	0	LDY	AØ	■	□	■	□	□	□	□	□
	1	"	AØ	■	□	■	□	□	□	□	□
	2	"	AØ	■	□	■	□	□	□	□	□
	3	"	AØ	■	□	■	□	□	□	□	□
	4	"	AØ	■	□	■	□	□	□	□	□
	5	"	AØ	■	□	■	□	□	□	□	□
	6	"	AØ	■	□	■	□	□	□	□	□
	7	"	AØ	■	□	■	□	□	□	□	□
	8	"	AØ	■	□	■	□	□	□	□	□
	9	"	AØ	■	□	■	□	□	□	□	□
	10	"	AØ	■	□	■	□	□	□	□	□
	11	"	AØ	■	□	■	□	□	□	□	□
	12	"	AØ	■	□	■	□	□	□	□	□
	13	"	AØ	■	□	■	□	□	□	□	□
	14	"	AØ	■	□	■	□	□	□	□	□
	15	"	AØ	■	□	■	□	□	□	□	□
USED FOR ALL LINE LENGTHS	16	"	AØ	■	□	■	□	□	□	□	□
	17	"	AØ	■	□	■	□	□	□	□	□
	18	"	AØ	■	□	■	□	□	□	□	□
	19	"	AØ	■	□	■	□	□	□	□	□
	20	"	AØ	■	□	■	□	□	□	□	□
	21	"	AØ	■	□	■	□	□	□	□	□
	22	"	AØ	■	□	■	□	□	□	□	□
	23	"	AØ	■	□	■	□	□	□	□	□
	24	"	AØ	■	□	■	□	□	□	□	□
	25	"	AØ	■	□	■	□	□	□	□	□
	26	"	AØ	■	□	■	□	□	□	□	□
	27	"	AØ	■	□	■	□	□	□	□	□
	28	"	AØ	■	□	■	□	□	□	□	□
	29	"	AØ	■	□	■	□	□	□	□	□
	30	"	AØ	■	□	■	□	□	□	□	□
	31	RTS	6Ø	□	■	■	□	□	□	□	□

6L-S64
PROM NUMBER

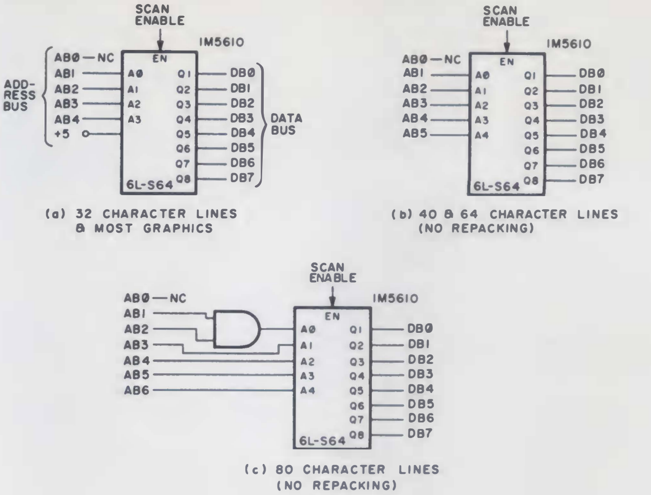
□ = "0"
■ = "1"
(POSITIVE LOGIC)

6502 CODING

USE FOR:
ALPHANUMERIC SCANS
• 32 CHARACTER LINES
• 64 CHARACTER LINES
• OTHER LINES THAT ARE NOT REPACKED

GRAPHIC SCANS
• 8/1 B/W
• 4/2 B/W
• 3/2 COLOR

Fig. 8a. Truth table for Scan PROM 6L-S64.



(a) 32 CHARACTER LINES & MOST GRAPHICS

(b) 40 & 64 CHARACTER LINES (NO REPACKING)

(c) 80 CHARACTER LINES (NO REPACKING)

Fig. 8b. Connections for Scan PROM 6L-S64. Scan enable input comes from instruction decoder.

can be a sequence of LDY commands followed by an RTS.

When a scan is wanted, the instruction decoder provides a ground on its scan enable output line. This ground is used to activate the Tri-state PROM that generates our scan microinstruction.

Once it is activated, this PROM takes over control of the computer's data bus. When not activated, the Tri-state outputs float and remain transparent to the data bus. This lets your computer behave normally during non-scan times. When you are scanning, it is very important that anything else that might want to use the data bus is disabled — this is what the DEN output on the instruction decoder is for. This output disables everything but the scan microprogram PROM when a scan is needed. The DEN output goes low for normal computer use and high for scan microprogram use.

Typical coding for a universal scan microprogram PROM is shown in Fig. 8a. Our code consists of 31 LDY commands followed by a single RTS. By ignoring address AO, we double this capability to get 62 microseconds' worth of "Load Y with the command for Load Y," followed by two microseconds' worth of advancing RTS.

This PROM coding can be used anywhere you want a scan of most any length, so long as memory repacking is not needed. Some typical connections appear in Fig. 8b. With input A4 positive, we can scan 32 or fewer characters per line. Any even number of characters is possible, but the packing density drops as you use fewer characters. This PROM is used in the TVT-6L for 32 alphanumeric characters, and in the TVT-7 for 32 chunks that result in 96, 128 or 256 horizontal graphics dots.

If all five inputs are used, we pick up a 34 to 64 character-per-line capability. This includes densely packed 64-character lines and non-repacked 40-character lines. Finally, if we add an external AND gate, we can go from 68 to 128 characters per line to pick up an 80-character, non-repacked ability. Line lengths with this gate must be some multiple of four.

Your turn: Show the PROM connections and memory map for densely packed lines of 8 and 16 characters.

Note that this PROM must have Tri-state outputs, since it's absolutely essential to float the outputs going to the data bus during non-TVT times. Note further that your coding will change with your change in microprocessor family.

INPUTS				OUTPUTS							
				Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
	WORD #	WHAT DOES THIS WORD DO ?	HEX OP-CODE	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
▨	0		AØ	■	□	■	□	□	□	□	□
SCAN "1"	1	LDY	AØ	■	□	■	□	□	□	□	□
	2	"	AØ	■	□	■	□	□	□	□	□
	3	"	AØ	■	□	■	□	□	□	□	□
	4	"	AØ	■	□	■	□	□	□	□	□
	5	"	AØ	■	□	■	□	□	□	□	□
	6	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "2"	7	LDY	AØ	■	□	■	□	□	□	□	□
	8	"	AØ	■	□	■	□	□	□	□	□
	9	"	AØ	■	□	■	□	□	□	□	□
	10	"	AØ	■	□	■	□	□	□	□	□
	11	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "3"	12	LDY	AØ	■	□	■	□	□	□	□	□
	13	"	AØ	■	□	■	□	□	□	□	□
	14	"	AØ	■	□	■	□	□	□	□	□
	15	"	AØ	■	□	■	□	□	□	□	□
	16	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "4"	17	LDY	AØ	■	□	■	□	□	□	□	□
	18	"	AØ	■	□	■	□	□	□	□	□
	19	"	AØ	■	□	■	□	□	□	□	□
	20	"	AØ	■	□	■	□	□	□	□	□
	21	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "5"	22	LDY	AØ	■	□	■	□	□	□	□	□
	23	"	AØ	■	□	■	□	□	□	□	□
	24	"	AØ	■	□	■	□	□	□	□	□
	25	"	AØ	■	□	■	□	□	□	□	□
	26	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "6"	27	LDY	AØ	■	□	■	□	□	□	□	□
	28	"	AØ	■	□	■	□	□	□	□	□
	29	"	AØ	■	□	■	□	□	□	□	□
	30	"	AØ	■	□	■	□	□	□	□	□
	31	RTS	6Ø	□	■	■	□	□	□	□	□

6L-S40
PROM NUMBER

□ = "0"
■ = "1"
(POSITIVE LOGIC)

6502 CODING

USE ONLY FOR 40 CHARACTER LINE REPACKED ALPHANUMERIC SCANS.

Fig. 9. Truth table for Scan PROM 6L-S40.

Fancier and more specialized PROM coding is needed if we are going to densely repack 40- or 80-character lines. Fig. 9 is the 6502 coding for a 40-character scan, while Fig. 10 is the coding for an 80-character scan. Both PROMs provide for repacking so that each page of 256 words has three 80-character lines or six 40-character lines.

Connections for either repacked PROM are shown in Fig. 11. An external three-input AND gate is used that lets us get the needed 128 equivalent words out of a 32-word PROM.

Your turn: Show how three switches or jumpers may be added to Fig. 11 to allow the same alphanumeric circuit board to work with any of the three scan truth tables shown.

Be sure to notice the difference in how the enable input is treated between the instruction decoder PROM and the Scan microprogram PROM. In the instruction decoder, the outputs must always be active, so we permanently enable this PROM. In the Scan microprogram PROM, we have to be able to Tri-state-float the outputs for all non-TVT times, and we drive the PROM's chip enable from an

INPUTS				OUTPUTS							
	WORD #	WHAT DOES THIS WORD DO ?	HEX OP-CODE	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
				DB7	DB6	DB5	DB4	DB3	DB2	DB1	DBØ
▨	0	LDY	AØ	■	□	■	□	□	□	□	□
	1	"	AØ	■	□	■	□	□	□	□	□
SCAN "1"	2	"	AØ	■	□	■	□	□	□	□	□
	3	"	AØ	■	□	■	□	□	□	□	□
	4	"	AØ	■	□	■	□	□	□	□	□
	5	"	AØ	■	□	■	□	□	□	□	□
	6	"	AØ	■	□	■	□	□	□	□	□
	7	"	AØ	■	□	■	□	□	□	□	□
	8	"	AØ	■	□	■	□	□	□	□	□
	9	"	AØ	■	□	■	□	□	□	□	□
	10	"	AØ	■	□	■	□	□	□	□	□
	11	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "2"	12	LDY	AØ	■	□	■	□	□	□	□	□
	13	"	AØ	■	□	■	□	□	□	□	□
	14	"	AØ	■	□	■	□	□	□	□	□
	15	"	AØ	■	□	■	□	□	□	□	□
	16	"	AØ	■	□	■	□	□	□	□	□
	17	"	AØ	■	□	■	□	□	□	□	□
	18	"	AØ	■	□	■	□	□	□	□	□
	19	"	AØ	■	□	■	□	□	□	□	□
	20	"	AØ	■	□	■	□	□	□	□	□
	21	RTS	6Ø	□	■	■	□	□	□	□	□
SCAN "3"	22	LDY	AØ	■	□	■	□	□	□	□	□
	23	"	AØ	■	□	■	□	□	□	□	□
	24	"	AØ	■	□	■	□	□	□	□	□
	25	"	AØ	■	□	■	□	□	□	□	□
	26	"	AØ	■	□	■	□	□	□	□	□
	27	"	AØ	■	□	■	□	□	□	□	□
	28	"	AØ	■	□	■	□	□	□	□	□
	29	"	AØ	■	□	■	□	□	□	□	□
	30	"	AØ	■	□	■	□	□	□	□	□
	31	RTS	6Ø	□	■	■	□	□	□	□	□

6L-S80
PROM NUMBER

□ = "0"
■ = "1"
(POSITIVE LOGIC)

6502 CODING

USE ONLY FOR 80 CHARACTER LINE REPACKED ALPHANUMERIC SCANS.

Fig. 10. Truth table for Scan PROM 6L-S80.

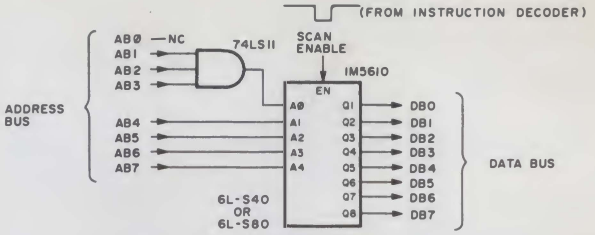


Fig. 11. Connections for 6L-S40 and 6L-S80 Scan PROMs that give densely repacked 40- or 80-character lines.

instruction decoder output, going low only when a scan is wanted. The instruction decoder PROM could be Tri-state, open collector or even permanently internally enabled, but the scan microprogram PROM *must* be Tri-state.

The instruction decoder PROM, the scan microprogram PROMs and possibly an AND gate or two are usually all we need to get a microprocessor outputting character or chunk words in a sequence and form that eventually will give us good video. The only signals fed back from the interface hardware to the microcomputer originate in these two PROMs. These signals are:

DEN: Decode Enable that goes high whenever a scan is to be produced and stops anything else from grabbing the data bus during scan times.

CSO: Chip Select Output that enables the display memory, either when the computer wants it for normal use or when the TVT wants to get characters out the upstream tap.

DB0-DB7: Data Bus outputs from the scan microprogram generator that are active whenever a scan is wanted, but Tri-state-floated otherwise.

The important thing to note is that these two PROMs plus any supporting gating always should be designed, tested and debugged first in any microprocessor-based video display. If they can't make the computer behave the way you want it to, nothing else you add in the way of interface hardware is going to work, either. ■