Don Lancaster
Synergetics

TVT Hardware Design

low-cost graphics

This is the second of a two-part series by Don Lancaster, taken from his new book The Cheap Video Cookbook *which will soon be published by Howard W. Sams. — John.*

The next most important block in our cheap video interface hardware is the data-to-video converter. The data-to-video converter receives code from the display memory's upstream tap and converts the input data into serial video. The input data can represent ASCII characters, hex op code or graphics chunks.

Graphics data-to-video conversion is usually simpler than alphanumeric conversion. For graphics use, we can sometimes get by with nothing but a shift register that converts the parallel chunk code into serial video. To this we might add an electronic selector to rearrange the chunk as needed for other formats. This selector can be a 4PDT switch that picks the upper or lower part of a chunk on a given scan.

For alphanumeric TVTs, there is no one-on-one relationship between the ASCII and cursor code stored in the display memory and the dots on the screen. Somehow, we have to irrationally fluff up our 6-, 7-, or 8-bit code into a 35- or 63-dot serial-video code. Since the character dots don't have any logical relationship to the ASCII code, any bits-and-pieces logic scheme is bound to be a complex disaster.

Instead, we go to the code conversion capabilities of a read only memory (ROM). You can use your own PROM for this, but code-converting read only memories called *dot-matrix character* generators are easy to get, usually cheaper and often a better choice. Details on these character generators appear in the *TV Typewriter Cookbook* (Sams 21313). Character generators can offer a choice of uppercase or combined uppercase and lowercase. They will either do the entire conversion to serial video by themselves, or else they will have multiple outputs that have to go to an external video shift register for final conversion.

For TVT use, your character generator must be of the *row-scan* type. There is another type called a *column-scan* character generator, but this is only good for strip printers, advertising signs and similar uses where the serial or parallel output runs up and down rather than back and forth.

An alphanumeric data-to-video converter using a 2513 character generator is shown in Fig. 12. The character generator accepts ASCII words from the upstream tap on the display memory. These ASCII words change once each microsecond for each new character to be output. The 2513 also accepts three "what line is it" commands from the instruction decoder. In exchange for these inputs, five dots are output at once, corresponding to one row on a 5 x 7 dot-matrix character. An eight-input, one-output shift register then converts these dots, along with spacing "undots" from grounded inputs, into raw serial output video. The input ASCII character coding repeats itself at least seven times to generate the entire seven dot rows involved in a row of characters. Our shift register is driven by a high-frequency timing circuit that outputs a narrow LOAD pulse once each microsecond, along with a CLOCK output that runs continuously at the desired dot rate.

An optional cursor is shown in the lower right of Fig. 12. The 4584 is a five Hertz oscillator that sets up the cursor winking rate. If ASCII input bit 8 is high, the CUR input will go high, and a white line is output on leads 01 through 05. The right diode causes this line to blink off and on, while the left diode allows winking cursors only during valid character times.

Since lowercase is not available, ASCII bit 7 remains unused. Note that if you want to display lowercase characters as uppercase, you must add a simple external gate, for the lower six bits of a lowercase u are the same as a 5, and not a capital U.

The 2513 is cheap and easy to get. The newer single-supply +5 volt versions by General Instruments and others are far easier to use than the old +5, ground -5, -12 versions. This is particularly important since all the

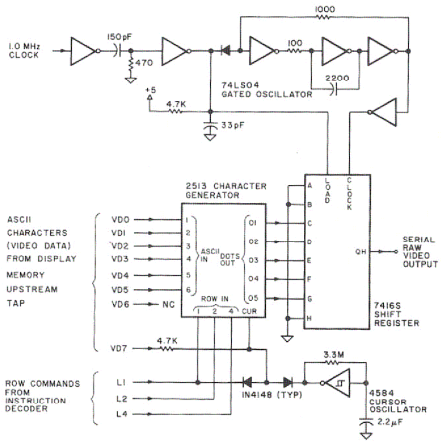

Fig. 12. Alphanumeric data-to-video converter using 2513. Circuit offers uppercase only; uses external video shift register.

rest of the interface hardware can run on a +5 volt supply. Lowercase versions of the 2513 are also available. You can use a pair of 2513s, one upper, one lower, for full-alphabet capability.

A premium 7 x 9 alphanumeric data-to-video converter using the Standard Microsystems CG5004 character generator is shown in Fig. 13. This circuit gives you both uppercase and lowercase and has its own internal shift register. It works on a single +5 volt supply. A winking underline cursor is produced automatically with the cursor circuit shown.

Your turn: Show how switching may be added to either 3-12 or 3-13 to give you manual control of cursor visibility.

Your serial-video output is called *raw video* because it contains only character dots when and where needed and blank logic zeros everywhere else. To get from here to something a TV set, monitor or rf modulator can handle, we have to add the sync pulses, and optionally predistort the raw video for improved clarity. We call everything compensated and combined *composite video*.

Fig. 14 shows us our first graphics interface. This circuit is used in the TVT-6 7/8 whenever eight dots per chunk in a row are needed. Input video data chunks from the display memory are routed directly to the eight inputs of a shift register. High-frequency timing applies just the right LOAD and CLOCK commands to output continuous dots during graphics display times. The pot in the timing is adjusted for minimum overlap or underlap between sequential chunks.

Fig. 15 shows us a graphics interface used for three or four dots per chunk output per row. Upper and lower chunk halves alternate for sequential lines or line pairs. A 74LS258 data selector is added to the inputs to pick upper chunk halves, lower chunk halves or blanking. We've pulled a trick here to introduce blanking. The ENABLE input to the 74LS258 drives all outputs *high* for blanking. To make this point in the circuit have *high* = black, we use an *inverting* data selector, and then pick the *complimentary* output of the shift register. We then end up with an input 1 giving us a white dot, an input 0 giving us a black dot and a blanking command also giving us a black output.

Your turn: Show a low-cost way (jumpers, switches, small module, etc.) to have both Figs. 14 and 15 use the same circuit board. Also provide blanking for Fig. 14.

Color can be added to the circuit in Fig. 15 with an external color modulator. The color format can be three dots on top of three dots, with the remaining two chunk bits letting us call any of four colors plus black.

High Frequency Timing

It's up to the high-frequency timing to give us the LOAD and CLOCK signals needed for serial output of video from the data-to-video converter. Traditionally, these circuits use crystal oscillators and counter-dividers for this job. For many microprocessor-based video display circuits, all we really have to do is use

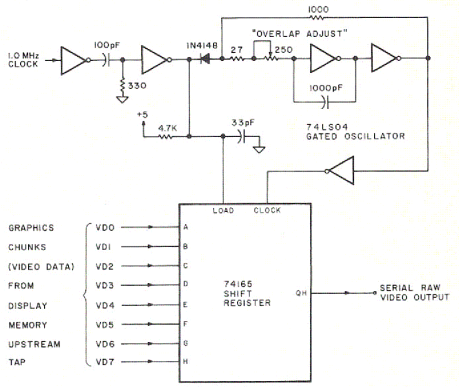

Fig. 14. Graphics data-to-video converter to display eight horizontal dots per chunk.

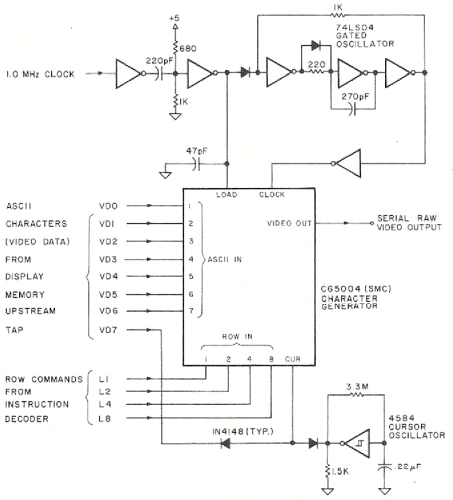

Fig. 13. Alphanumeric data-to-video converter using 5004. Circuit offers uppercase and lowercase; has internal video shift register.

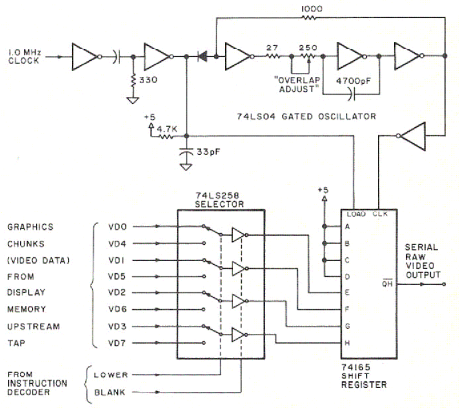

Fig. 15. Graphics data-to-video converter to display three or four horizontal dots per chunk on alternate line pairs.

a plain old one-megahertz microprocessor clock and add a simple gated oscillator using a hex inverter to get our LOAD and CLOCK waveforms.

Four examples of these hex-inverter high-frequency timing circuits appeared on the tops of Figs. 12 through 15. Fig. 16 tells us what these timing circuits have to do, while Fig. 17 shows us typical waveforms.

The LOAD output of the timing circuit has to transfer parallel dots into the video shift register. *This LOAD pulse must be carefully timed to arrive only when data is ready and settled from earlier portions of the data-to-video converter*. Often, you will have two choices of one-megahertz clock available; if one phase doesn't do it, the other one probably will. Most often, it's best to arrange the load command so it always arrives one microsecond after the address change. This gives you a nearly maximum processing time and minimizes any settling or bad-data problems. Note that the cursor and blanking should be introduced *before* this one microsecond delay takes place; otherwise they will also have to be delayed somehow.

The polarity of the LOAD output has to match whatever is doing the final serial conversion. A 74165 shift register needs a normally high LOAD command that briefly goes low for loading. The CG5004 needs the opposite. In general, LOAD command pulses should be as narrow as possible. This is particularly true in graphics interfaces where a too long or misplaced load pulse can cause dot overlap or underlap.

Your CLOCK line decides how fast the dots are going to come out as serial video. In an alphanumeric display, too low a dot rate will cause the characters to overlap, while too high a dot rate will make the characters narrow and will also need too much video bandwidth. Between these limits, the frequency of the clock gives us control of the spacing between characters. The CG5004 is very fussy over its clock duty cycle. It demands the narrow positive pulses shown in Fig. 17b.

In a graphics display, the dot rate sets how many dots you get per chunk of input data. Fine tuning the dot rate is usually necessary so that the end dot of one chunk appears the same size as the start dot of the next chunk. A frequency error here will cause dot overlap or underlap.

CLOCK and LOAD must be locked together to prevent the dot locations from jittering or otherwise smearing. It's also particularly important to make sure the LOAD command does not distort the clock on graphics displays; otherwise clockings end up wider or narrower with respect to each other.

The top half of Fig. 12 shows a typical hex-inverter high-frequency timing circuit. The first inverter acts as a buffer to make us independent of system clock rise and fall times. The second inverter is a half monostable, whose output briefly drops to ground for 60 nanoseconds on the falling edge of the clock input. The three upper-right inverters are a gated ring oscillator running at a seven-megahertz rate. Frequency is set by the 100 Ohm resistor and 2200 pF capacitor. Gating is done by the diode. This synchronizes the oscillator to the LOAD command every time the LOAD command goes low. A final buffer and inverter are used to square up and invert the clock line. The 4.7k and 33 pF network is a glitch filter for added stability.

The high-frequency timing circuits are about the same, differing only in speed and polarity details.

Most often, some cut-and-try is needed in getting a hex-inverter high-frequency timing circuit to work the first time. But, once you have a working circuit, it is usually tolerant of normal production component spreads. Fine tuning is usually needed in graphics applications to control overlap and underlap, besides giving a handy way to change from three to four dots per chunk. Usually fine tuning can be omitted on alphanumeric uses as the main effect of slight frequency changes is a small change in character spacing.

Your turn: Show a way of raising and lowering the microprocessor clock frequency to allow locking of the video display's vertical rate to the power line. Can a

---

the

HIGH
FREQUENCY
TIMING

must

* Deliver a "load" pulse to the video shift register.
* Time the "load" pulse to arrive only when data is valid.
* Deliver "clock" pulses to set the video dot rate.
* Lock "load" and "clock" pulses together to prevent jitter.

Fig. 16. High-frequency timing traditionally has used crystals and divider chains, but a hex-inverter gated oscillator is often all that is really needed.

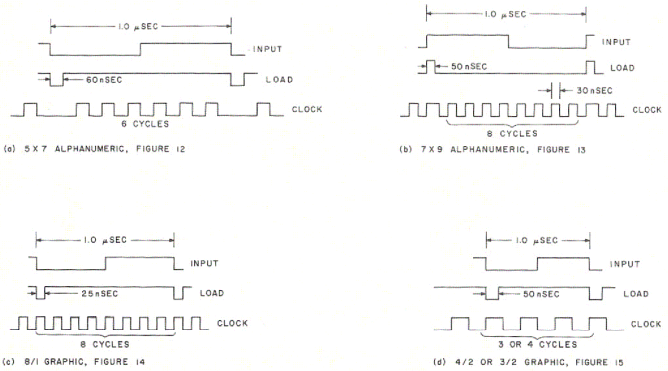

Fig. 17. Typical high-frequency timing waveforms.

CMOS 4046 be of help here?

As a rough rule of thumb, the output video frequency will be around one-half the dot rate set by the high-frequency timing clock. Thus, eight dots per microsecond gives around four megahertz bandwidth, while three dots per microsecond needs only a megahertz-and-a-half bandwidth. A black-and-white television set has a four megahertz video bandwidth, extendable somewhat by defeating the sound trap. The video bandwidth of a color set is limited to three megahertz. As you can see, the output frequencies associated with microprocessor-based video displays are compatible with TV sets with largely unmodified video bandwidths. This is a dramatic improvement over the 10- to 20-megahertz bandwidth often needed by traditional video terminal systems.

Sync and Position

When we run a properly debugged scan program, our instruction decoder will give us signals at the needed vertical (V SYNC) and horizontal (DEN) rates. We can then selectively delay these signals to gain control over position. This is followed by pulse shaping to get the proper widths of the sync signals for TV use. Since the TV set locks itself to the sync pulses, backing up or moving these pulses forward has the effect of moving the entire display. Horizontal delay changes cause back-and-forth position changes in the display, while vertical sync changes control up-and-down positioning.

A typical circuit is shown in Fig. 18. Once again, it is six inverters to the rescue. However, this time two of the inverters have to have very low open collector output impedances and are TTL, while four of them need extremely high input impedances and a snap action, so they are CMOS Schmitt inverters.

Our V sync pulse starts out as a one microsecond positive pulse that is glitch-filtered to get rid of anything that crops up during the PROM settling times. The 7405 discharges the 4700 pF capacitor completely once each 1/60th second. This capacitor is recharged by a rate you set with the V POS control. When the recharging reaches around one-half the supply voltage, the Schmitt snaps on, and our second stage output becomes a square wave delayed by the amount set on the position pot. Because of the extreme differences in charge-to-discharge times of the capacitor doing the positioning, the TTL/CMOS combination is called for. The delayed output is shaped into a positive-going 200 microsecond pulse by the final Schmitt and RC network. The output resistor aids in interfacing the TTL stage that follows in the video output circuitry.

The horizontal circuit is similar, with only the timing details changing. The DEN output of the instruction decoder can often be used, instead of needing a special H SYNC line. Delay of a portion of the horizontal line is done with the first variable RC network, while the second RC combination gives us a five microsecond sync pulse once every horizontal line.

This particular sync and position circuit needs continuous arrival of H and V signals from the instruction decoder. This continuous need limits the transparency and throughput of the computer on other programs that are also active while the TVT is displaying.

Fig. 19 shows us a different way to get horizontal sync pulses. This counter method can free the computer for other uses during vertical retrace times. This in turn can greatly increase the transparency and throughput.

What we have is a divide-by-H counter. H is set to the number of microseconds per horizontal line. For every overflow, an H sync pulse is delivered, regardless of what the computer happens to be doing. This counter is synchronized to the scan program by resetting it with the instruction decoder. The synchronization and reset can take place on every active line, once during vertical retrace, or even only once during power up. Horizontal sync is maintained through vertical retrace times, even if the computer is busy working on something else.

Bandwidth Compensation and Video Output

We now have three signals available — raw video, vertical sync and horizontal sync. We somehow have to combine these and pick up some line

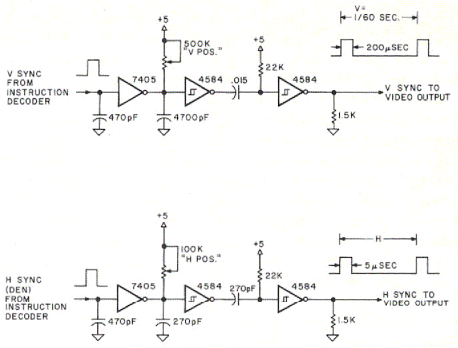

Fig. 18. Sync and position circuitry.

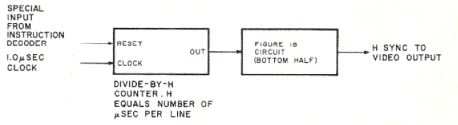

Fig. 19. Modified horizontal sync and position circuit gives high transparency and throughput.

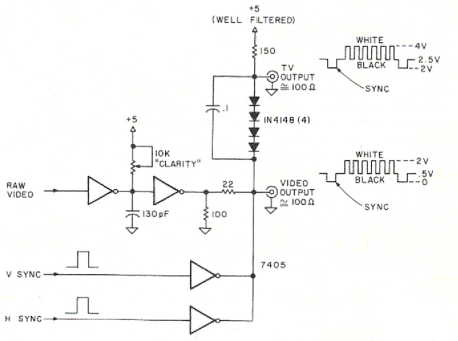

Fig. 20. Bandwidth compensator and video output circuit. Video output is for monitors and rf modulators. TV output is pretranslated for minimum set modifications.

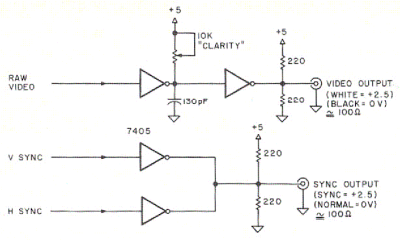

Fig. 21. Alternate bandwidth compensator and video output circuit for monitors with separate sync input.

drive capability if we are going to interface a TV set, monitor or rf modulator. This final interface is the purpose of the video output circuitry.

Our raw video first goes to a *bandwidth compensator*. This super-important circuit tries to anticipate how the TV set is going to degrade the response, and then predistorts the video in the opposite direction beforehand. You do bandwidth compensation by making the dots longer than the undots. One way is to OR the raw video with a delayed replica of itself. A simpler but very sneaky way is shown in Fig. 20. An open collector TTL inverter has a much lower output low ON impedance than its output high OFF impedance. If we add capacitance from this output to ground, the capacitor will discharge fast, but its charge rate will be much slower and set by the value of the pullup resistor, which in this case is a CLARITY pot. Since this is an inverter, a white dot is low and a black undot is high at the capacitor. It takes longer to get out of the low state, so our dots automatically get lengthened.

The amount of lengthening is set by the CLARITY pot. This pot is adjusted for the densest, clearest characters on the final TV screen. The optimum setting is often the one that just barely closes the inside of an M or a W on the display. The use of this bandwidth compensator and our one microsecond constant character or chunk time are the two keys to display of quality characters or graphics on a TV set with unmodified video bandwidth.

Three more open collector inverters are used for video combination. At the VIDEO output, sync pulses are nearly at ground, while black is at 0.5 volts and white is near +2 volts. The ratios of these three values are set by the three resistors. This output is essentially a standard form for video monitors, rf modulators and TV sets that have been completely converted internally for direct video monitor use.

But, we've also provided a new TV output, which has the same waveform but is translated up so that white is at +4 volts, black at +2.5 volts and sync at +2 volts. The +4 white level is the normal bias level at the video detector of most solid-state TV sets. You can often use this TV output to go directly into the first video stage of many TV sets, without needing anything else in the way of translation or bias circuits.

Some monitors have separate VIDEO and SYNC inputs. These are called split sync systems, and an alternate dual output circuit shown in Fig. 21 may be used if split sync is needed or wanted. ■