TV Color Graphics

Don Lancaster
Synergetics

A graphic display lets us add pictures, lines, and symbols to the basic letters and numbers of an ordinary TV typewriter so that it can be used in chess, for video art, for games, printed circuit layouts, logic diagrams, charts, and so on. For many TVT and microcomputer uses, the ability to display several colors at once is very important; for others, it is simply a convenience that may not be worth the extra complexity.

Dots or Vectors?

There are two fundamentally different approaches to graphic displays. One is called the stroke method. The display is told where to start each symbol, which direction it should take, what color to display, and how far to go before stopping or changing direction. On the other hand, dot matrix method breaks the display down into as many raster scanned boxes or dots as a display refresh memory can provide. These blocks are lit or not lit with a selected color as needed to form an image.

The stroke method somehow sounds better, particularly if we are interested in curved figures and smooth edges on diagonal lines. But it has problems. Its most serious shortcoming is that it is not directly compatible with either a stock black and white or a stock color TV set. A special display is thus needed, although an oscilloscope can be substituted for single color, low resolution use. Another limitation is that the display and memory bandwidths have to be extremely wide, or else the display has to slow down its refresh rate as more and more symbols are added to the display. This can lead to unacceptable flicker even with a special tube phosphor.

The raggedness of the dot matrix system can be minimized by using enough memory to increase the resolution, and by minimizing the curved or diagonal lines to be displayed. The dot matrix technique has one major advantage in that we can easily and efficiently mix graphics and alphanumerics, simply by using a TVT system in a graphic mode on one field and in an alphanumeric mode on the second, merging the two separate displays as one. Besides giving more pleasing characters, this approach has an efficiency advantage – only six bits are needed to store a character, but 35 or more bits are needed to store and display the extended dot matrix character in a 5 by 7 format.

It turns out that a dot matrix display for graphics only is actually easier and simpler to build than an ordinary TVT; the character generator is no longer needed, and the high frequency timing is usually very much simplified. This type of display lends itself very well to existing microprocessor organization without extensive interface.

One limitation of the graphics dot matrix approach is that quite a bit of memory is needed, one bit corresponding to each and every possible dot location.

Some Television Limitations

What are the limits to the matrix size we can display on a television set? First and foremost, this depends on the amount of memory available because one bit will be needed for each dot position, and a fractional part of a bit (more on this later) will be needed for that location's color information. If we are also interested in *gray scale* (brightness variations), more than one bit per dot position will probably be needed. Gray scale techniques may not be worth the added cost and complexity.

Black and white sets can allow very dense matrices. For instance, if we have a black and white set modified to display 72 characters in a 7 by 9 dot matrix with two vacant dot positions between characters, we end up with a horizontal resolution of 648 dots over the active scan. (This takes direct video and some modifications.) The scan rate limits our dots vertically. Without interlace, we get 262, 262.5 or 264 dot resolutions. If we have to, we can go to full interlace and double these values. Thus, it is quite possible to have a 512 by 512 dot graphic display. The memory for this display would take 262,144 bits, or 32,768 eight bit bytes. At 0.1 cent per bit, such a memory costs around $264. Besides being expensive,the density pushes the line pair resolution of the TV. Black and white sets can handle just about anything we can afford to throw at them in the way of graphics.

Color sets are much more restrictive, but still give us more than enough capability for very useful display matrices. An unmodified color TV has a maximum video bandwidth of 3 MHz brought about by the need to eliminate a 3.58 MHz subcarrier in the color processing video. Three MHz video is roughly equal to 6 million dots per second, which gives us around 380 or so dots per line. This doesn't include an allowance for horizontal retrace, so approximately 256 dots horizontally is a good limit. Similarly, if we don't use interlace, 256 dots vertically is also a reasonable limit. A 256 by 256 display contains 65,536 dot locations and costs us around $65 for a single color display at 0.1 cents per bit, and somewhat more if multiple colors are to be added.

Even if we were to separate totally the luminance and chrominance channels on the color TV (A major job that eliminates use of the TV for anything else), we'd still be limited by the number of holes in the shadow mask to resolutions only somewhat better than we can get on an unmodified color TV.

One very important thing to note is that the color bandwidth is far less than the dot resolution on a color TV set. In addition, the color bandwidth varies with the colors in use. The available bandwidth ranges from a high of 1.5 MHz for an orange cyan display to a low of 0.5 MHz for a red blue green tri-color display. Thus, with a high resolution display, it is not possible to change the color of each and every successive dot in the horizontal direction. For full color presentation, something less than 50 color changes across the screen is pretty much an upper limit, particularly if we are to minimize fringing and edge shading effects and are using saturated colors.

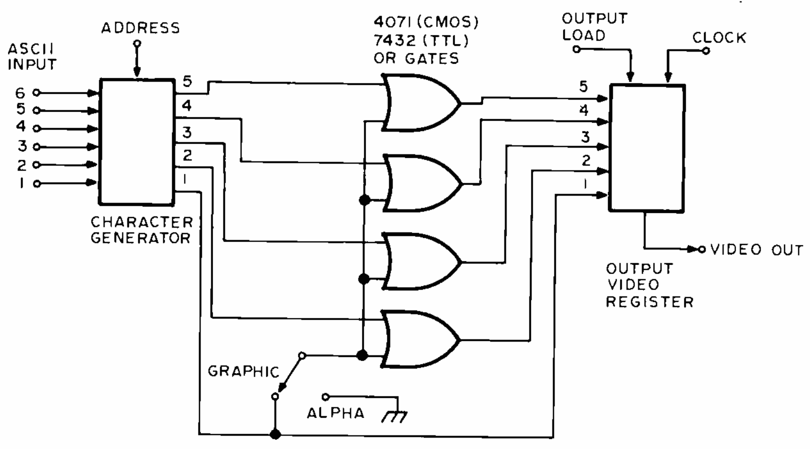


Figure 1: An ordinary alphanumeric television typewriter can be converted into a simple (but limited) graphics display by adding four OR gates as shown here.

The limited color bandwidth is both good and bad for graphics displays. The lower horizontal color resolution means that colors should remain constant for several dot positions at a time in the horizontal direction and whenever possible, we should change colors on a black dot or blanked portion of the display. Since this is the best we can do we use only a fraction of a bit per dot to specify the dot color. This means that our color storage won't add much to the basic memory size and cost. For instance, on an eight bit word, two color bits can specify any of four colors to six different dot locations. Or one color bit could specify a choice of two colors to seven different dot locations. Fortunately, black dominates and counts as wide bandwidth luminance video rather than color, so we can shift between black and colors and back again at full tilt.

Some Simple Graphics Displays

An ordinary alphanumeric TVT can be converted into a very limited performance graphics display simply by adding one quadruple OR gate package as shown in figure 1. The OR gate goes between the character generator ROM outputs and the output video shift register. It monitors the leftmost dot output and makes the other dots 1 if the leftmost dot is a 1. If you type a space, you get a blank location. If you type a D for a dot, you get a white box at each and every character location specified. There's nothing magic about using a D; any character with all leftmost dots which are a 1 will do, such as B, E, F, H, K, L, M, etc. . .

The resolution of your display depends on the original format, for you get one box

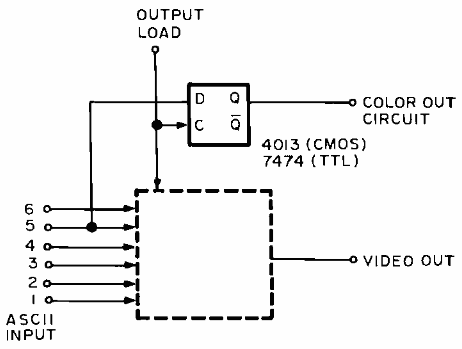


Figure 2: By using one of the six input lines to control the state of a flip flop, a two color display can be achieved. Use the color out line to drive the color modulator.

per character. With a 32 character per line display of 16 lines, you get a total of 512 "black surround" boxes in your matrix. While this isn't very attractive or much in the way of resolution (What do you want for only 14¢ extra in circuitry?), it is handy for studying graphics techniques, and is of limited use in games, displays, simple video art applications, and for generating large alphanumeric characters.

Figure 2 shows how to add color. Since we used only one bit of the word to store the alphanumeric information, we have enough bits left to handle quite a few different colors, ranging from 32 colors on a six bit memory word to 128 on an eight bit memory word. The color information is removed directly from the character storage memory, monitoring bit 5. The color output is delayed and resynchronized to the output video with the flip flop as shown. This delays the color to match the delay involved in the character generator. The color flip flop's output goes to a color subcarrier generator. The subcarrier is programmed to produce a red output when the fifth bit is a 1, and a blue output when it is a 0. It returns to reference phase during horizontal retrace.

Now, for no box, we type a space. For a red box, we type a R and for a blue box, we type a B. The first bit of the character generator *output* decides box or no box (luminance), while the delayed fifth bit of the character generator *input* decides red or blue.

We can pull a sneaky trick to double the resolution in the vertical direction, by making the top half of any box one color and the bottom half a second color. Better yet, we blank out enough of the middle of the box to exactly match what's blanked out *between* vertical boxes, so we get even spacing of all the boxes in the vertical direction. For instance, on a 5 by 7 dot matrix with three lines between vertical characters, we use only the two upper dots and the two lower dots, skipping the three middle ones.

Figure 3 shows the details. We carefully search the dot patterns of all the displayed characters to come up with some useful combinations. The inverse of bit one output is what counts in the double mode. By typing a B we get neither box in a character location. Typing an A gets us an upper box but not a lower box. A V gets us a lower box but not an upper one. And a < gives us both upper and lower boxes. Color may be added using the circuit of figure 2 connected to the seventh and eight memory bits.

A 96 by 96 Full Color Display

Let's look at the design problems involved in building a full color display that uses a modest memory of 2048 by 8 bits for storage (costing $16.34, at 0.1 cents a bit) and yet still has enough useful resolution for chess displays, Pong style games, video synthesis, creative art, and so on.

The central problem in designing a graphic display is proper *partitioning* of the memory. Partitioning simply decides what portion of the memory is related to the display in what manner. The three very important partitioning questions are:

1: How many resolvable elements in what positions and colors are supplied by a single memory word?

2: How is graphic information separated from alphanumeric information such as scores and printouts?

3: How do we separate the "seldom changing" memory portions of a display (such as the chess grid or a foul line) from the "often changing" parts (such as a just captured rook or a fast moving hockey puck) to simplify as much as possible the software and external control of the display?

For a 96 by 96 color display useful for chess and Pong style games, we might partition each memory byte as shown in figure 4. The first six bits of our eight bit

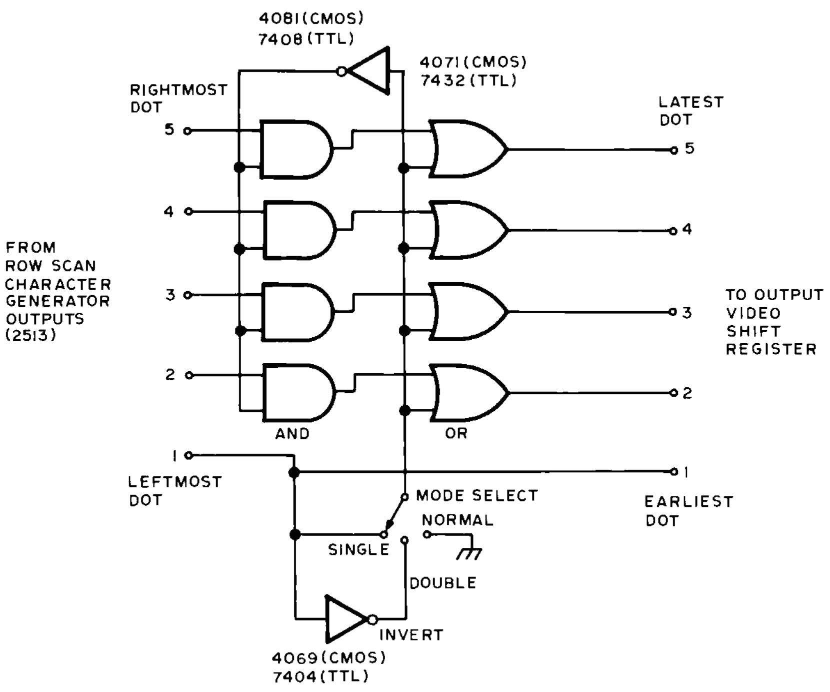


Figure 3: Logic required to generate two possible boxes within one character position of a TV Typewriter.

word specify the luminance (black or not black) of six dot locations; the final two bits specify the color of all six of those locations simultaneously. This gives us up to four colors plus black, and lowers the color change resolution enough that the color TV stays happy. In a Pong style game, the goal and sidelines can be one color and the opposing teams two additional colors, and the ball or marker yet another color. For chess, we can use a two-color chessboard with two-color opponents.

While our first six bits could be arranged as six successive dots horizontally or vertically, a grouping of three horizontally by two vertically seems to work out well for

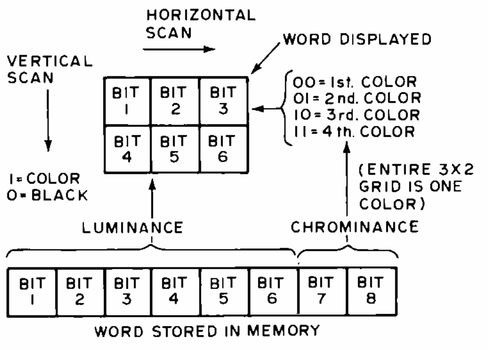
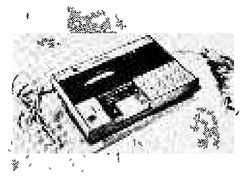


Figure 4: Using one byte of memory to specify a 3 bit segment of two horizontal scan lines, with one of four colors.

DIGITAL DATA RECORDER for Computer or Teletype Use
Up to 2400 Baud $149.95



Uses the industry standard tape saturation method to beat all FSK systems ten to one. No modems or FSK decoders required. Loads 8K of memory in 34 seconds. This recorder enables you to back up your computer by loading and dumping programs and data fast as you go, thus enabling you to get by with less memory. Great for small business bookkeeping. IMAGINE! A year's books on one cassette.

Thousands are in use in colleges and businesses all over the country. This new version is ideal for instructional, amateur, hobby and small business use. Ideal for use by servicemen to load test programs. Comes complete with prerecorded 8080 software program used to test the units as they are produced. (Monitor)

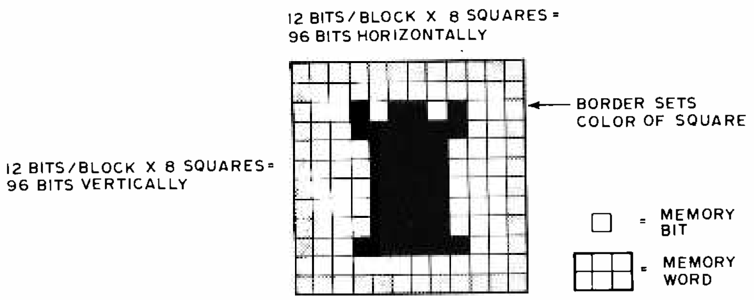
SPECIFICATIONS:
A. Recording Mode: Tape saturation binary. This is not an FSK or Home type recorder. No voice capability. No modem.
B. Two channels (1) Clock, (2) Data. Or, two data channels providing four (4) tracks on the cassette. Can also be used for NRZ, Bi-Phase, etc.
C. Inputs: Two (2). Will accept TTY, TTL or RS 232 digital.
D. Outputs: Two (2). Board changeable from TTY, RS232 or TTL digital.
E. Erase: Erases while recording one track at a time. Record new data on one track and preserve three or record on two and preserve two.
F. Compatability: Will interface any computer using a UART or PIA board. (Altair, Sphere, M6800 etc.)
G. Other Data: 110 V - (50-60) Hz; 2 Watts total; UL listed #955D; three wire line cord; on/off switch; audio, meter and light operation monitors. Remote control of motor optional. Four foot, seven conductor remoting cable provided.
H. Warrantee: 90 days. All units tested at 110 and 2400 baud before shipment. Test cassette with 8080 software program included. This cassette was recorded and played back during quality control.

ONE DOLLAR brings operating and maintenance manual with schematics.

COMING NEXT MONTH – IN KIT FORM
* Hexadecimal Keyboard – Load programs direct from keyboards' 16 keys and verifying display. Does not use Computer I/O.
* I/O for use with Computer Aid or other digital recorders. Variable baud rate selectable on externally located unit by one knob. Can load computer or accept dumps without software. Turnkey Operation. For any 8 bit computer.
* Record/Playback Amplifier. Expanded version of our Computer Aid board for use with your own deck (cassette or reel to reel). Go to 9600 baud on reel to reel. Digital in, digital out, serial format.
* Interested in these? Send your name and address for brochure when released.

Send check or money order for $149.95 plus $2.00 Shipping & Handling (N.J. Residents add 5% Sales Tax) to:

NATIONAL multiplex CORPORATION
3474 Rand Avenue, Box 288
South Plainfield, New Jersey 07080



12 BITS/BLOCK X 8 SQUARES = 96 BITS HORIZONTALLY
12 BITS/BLOCK X 8 SQUARES = 96 BITS VERTICALLY
BORDER SETS COLOR OF SQUARE
= MEMORY BIT
= MEMORY WORD

Figure 5: Setting a chess piece into a grid of colored squares of a color graphics display. For the chess board application, a grid of 8 by 8 = 64 squares is required.

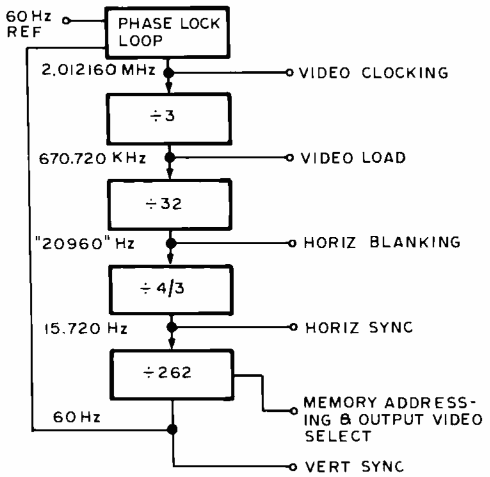
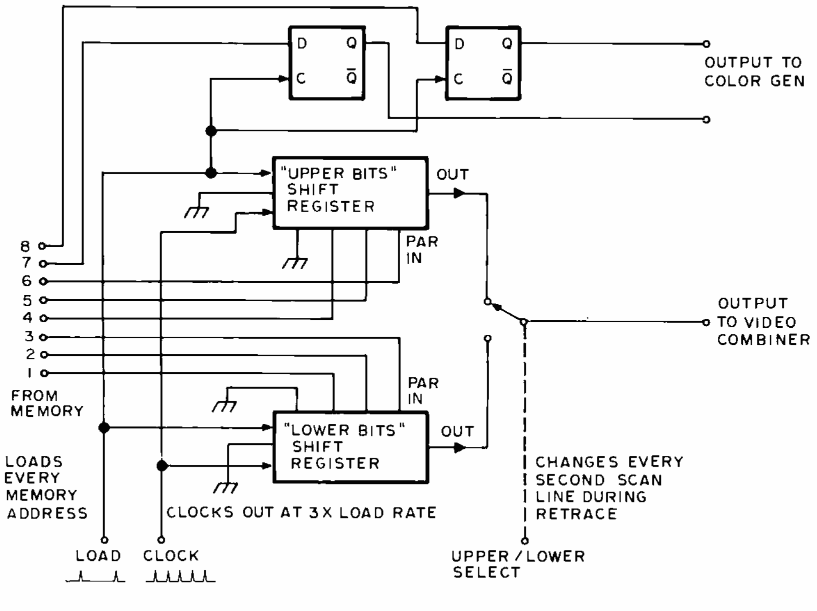


Figure 6: System timing for a 96 by 96 display grid. In this scheme, the phase lock loop is used to maintain synchronization with the 60 cycle line frequency.



D Q
C Q̄
D Q
C Q̄
OUTPUT TO COLOR GEN
"UPPER BITS" SHIFT REGISTER
OUT
PAR IN
8
7
6
5
4
3
2
1
FROM MEMORY
OUTPUT TO VIDEO COMBINER
PAR IN
"LOWER BITS" SHIFT REGISTER
OUT
LOADS EVERY MEMORY ADDRESS
CLOCKS OUT AT 3X LOAD RATE
CHANGES EVERY SECOND SCAN LINE DURING RETRACE
LOAD
CLOCK
UPPER/LOWER SELECT

Figure 7: The video output circuitry for a graphic display is simpler than that of a character display. This logic implements the 3 by 2 format of figure 4.

chess and many other games. The basic chess grid is shown in figure 5 and uses one eighth of the 96 elements in both directions. The choice of a rectangular grouping is also handy in games, for the ball or puck can be double size in both directions compared to the goal and foul lines and still be specified with only one memory word.

Having picked our basic format, we can go on and design our system timing. We start with a vertical rate of 60 Hz and use a 262 line non-interlaced system, for a total of 192 active scan lines and 70 retrace and blanking lines in the vertical direction. Thus two vertical lines are used for each resolution box, and 96 * 2 = 192.

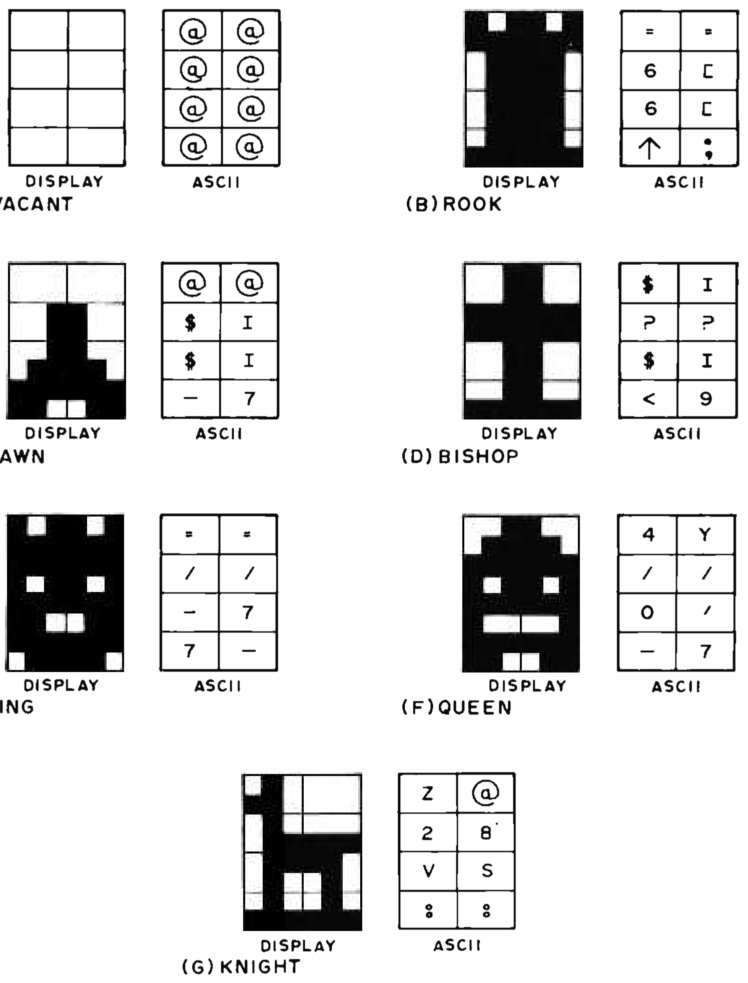
Our horizontal frequency will be 60 * 262 = 15,720 Hz, equal to a 63.6 microsecond horizontal line. With a reasonable horizontal retrace, this gives us around 48 μs of live scan time for 96 elements. The time per element is thus a relatively lazy half microsecond. There are three display elements for each horizontal word, so memory clocking will be needed at 1.5 μs intervals. One possible system timing setup is shown in figure 6.

The output video circuitry is simpler than in an alphanumeric display since we don't need a character generator. Figure 7 shows one possible arrangement. The memory words are clocked once every 1.5 μs and three bits are loaded into a four bit shift register and clocked out at a 2.01216 MHz rate. The other three memory bits are loaded into a second identical shift register. At the same time, the two color bits are decoded and sent to the color subcarrier system.

The registers are selected on alternate scan lines. This automatically picks the first three bits for the upper three boxes and the second three for the lower three boxes of each memory. Note that we can use a data selector since we only change it during the horizontal retrace time. The whole job can not be done with selectors due to the glitch and settling time problems. Note also that we get the same memory words back in the same sequence for *four* consecutive scan lines, two to display the first three bits and two to display the second three bits. This is similar to the readdressing that takes place on an alphanumeric TVT when we get each horizontal line back at least seven times to produce an entire dot matrix character.

The output of the data selectors is a video brightness signal which is routed to a typical video combiner. The color information is routed to the subcarrier circuit. Be sure that the color circuitry returns to the reference phase during the horizontal retrace time. Try to design the display so that most color

Figure 8: A set of chess men constructed using the video output circuitry of figure 7: The diagram shows a 6 bit ASCII subset character pattern to the right of each figure to show what should be loaded into the memory.



changes take place on a black or blanked location. If color fringing becomes a problem, a slight additional amount of video delay (with respect to the color changes) may minimize this effect.

Using this circuitry, each chess piece requires 8 characters of memory. Figure 8 shows one possible arrangement of the chessmen for a chess display (along with 6 bit ASCII subset character codes). The program of figure 9 shows us the starting position for a chess game. A basic hockey or Pong setup is shown in figure 10 with its program.

We can add scoring or other alphanumerics most simply by using an ordinary TVT with the graphics display and alternating frames, as shown in figure 11A, or by summing video as shown in figure 11B. Both system timing circuits must be locked together and have the same number of horizontal lines, and the same amount of horizontal retrace must be provided in both modes if the television is to recognize a continuous program. Another possibility is to use simple box "stroke style" characters and store them directly in the memory matrix.

Figure 12 shows several approaches to the third partitioning problem, the separation of often and seldom changing data. Most often, a graphics TVT display will be used in association with a microprocessor to get extensive game capabilities. This has a tremendous advantage over wired logic in that we can change rapidly a game format simply by changing programs.

The object of memory partitioning is to trade off specialized timing against the total number of memory bits that have to be involved with every computation and up-

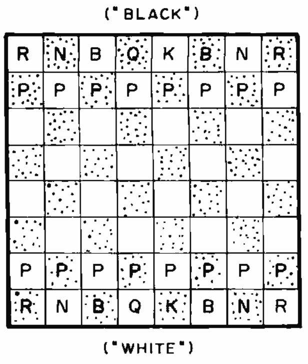
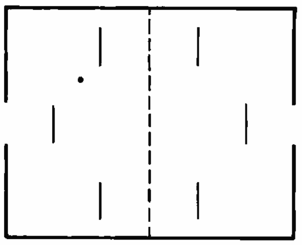


Figure 9: The starting configuration of a chess game is established by repeating the appropriate character patterns at selected places in the display memory and surrounding the piece patterns by the border pattern. If the "∘" character is used to represent the 8 elements of the chess man selected from figure 8, then the pattern for each piece is:

```
O G G '
C ∘ ∘ $
C ∘ ∘ $
C ∘ ∘ $
C ∘ ∘ $
C ∘ ∘ $
9 8 8 <
```

Each "∘" should be replaced by a character from one of the chess patterns of figure 8. Since the border pattern is fixed, moving a chess piece is accomplished by moving only the 8 characters in the center of each board position. Software for a chess display would initialize the border pattern at the start of play.

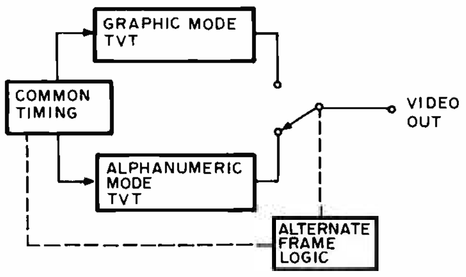


```
O G G G   G G G G   G G G G   G G G V   G G G G   G G G G   G G G G   G G O @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @

I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   [ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @

I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
```

Figure 10: Display pattern for a Pong style video game. This figure shows the contents of display refresh memory for the upper half of a playing field display. The lower half is a mirror image except for the location of the ball (which would be controlled by the software of your microcomputer.) The characters shown are 6-bit ASCII subset characters which give the proper codes for the 3 by 2 format of the video graphic generator of figures 4 and 7.



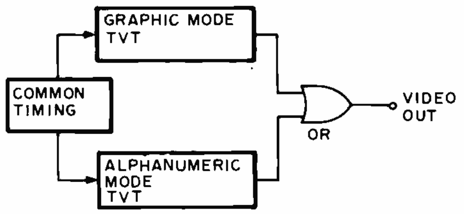


Figure 11: Combining a graphic display generator with a character display generator for a single television set. One way is to alternate on every other frame using a digital switch. A second method is to sum the video signal (using the OR gate) producing a combination output.

date. For instance, in figure 12A, we simply use the entire 2 K by 8 bit memory in one brute force piece. While conceptually the simplest and most flexible, it takes extensive software and a long time to fill the display initially and then to specify any changes.

In figure 12B, we have added some dedicated timing that separates the chess playing spaces from the board spaces. This is done with some "crosshatch" logic gates that pick out the places the chessmen are to go. Now a much smaller portion of memory is involved in specifying piece locations. While figure 12A requires 1536 words, the design of figure 12B necessitates 512 words.

Can we do better? In figure 12C, we've broken our memory into three distinct areas and selected them with wired logic. The main area is the board background which can be a ROM or a RAM which is loaded once and never changed. The second area is a file where we have stored the programs for the six chessmen involved (pawn, bishop, knight, rook, king, and queen). This file can repeatedly be addressed to go in different locations in the main graphics display.

Finally, we have a small active memory that decides what chess piece of what color (or the option of no piece) goes in what location. This active portion of our memory can be as little as 32 words since there are only 64 squares and only four bits are

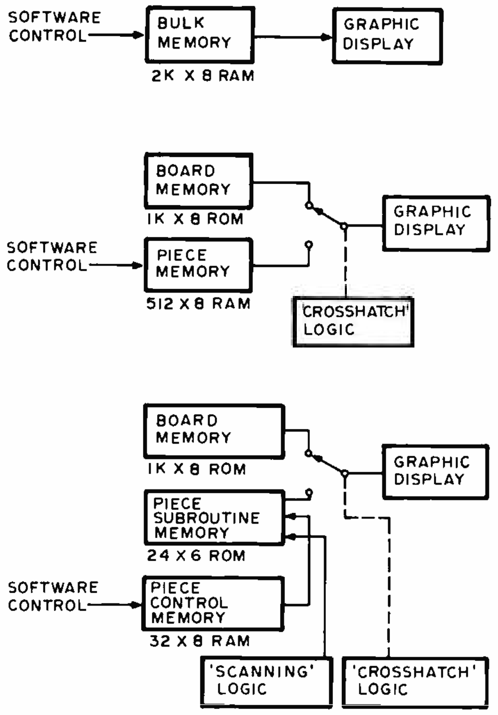


Figure 12: Three ways of organizing the refresh logic. At A, the most general approach is taken, with a full refresh memory for the entire screen. In this approach, computer software completely controls placement of patterns. At B, the memory is reduced slightly by partitioning it into two segments: One segment has the fixed "board layout" which is set up at the start of a game (or burned into the ROM of a commerial product); the second segment has the varying information on placement of the game pieces, and a crosshatch switch electronically combines them. Finally at C, even further specialization is introduced with three segments. The board memory remains, as in option B. Each piece of a chess grid is presented as a special ROM which is fixed with the bit patterns needed to generate any one of the 6 possible chess men. The placement and color are specified by a final RAM segment which changes during the game.

needed to specify any chessman or blank of either color. Here we trade the simple generality of a 2 K by 8 refresh memory off against the specific optimization of the 3-tiered memory for board games. Only a small amount of dedicated logic is needed for this specialization.

In a Pong style game, the ball is specified as four bits of the six bit word, making it twice the size of the background, foul, and goal lines. Ball motion can usually be computed during the vertical retrace time. For instance, if our microprocessor has a 10 microsecond cycle time, and we have around 4 milliseconds of vertical retrace time, up to 400 machine cycles are available to compute the new ball location.

The 96 by 96 format seems to lend itself well to game graphics and seems easy and reasonable to build. It is by no means the only possible organization for graphic displays. The basic partitioning problems will change for different types of graphic displays.■

NUMBERS

Robert Baker
34 White Pine Drive
Littleton MA 01460

Hidden in the matrix are 39 words related to numbers. Find a word, circle it or color it in, and cross it off the list. Words may be forward, backwards, up, down, or diagonally but always in a straight line; never skipping letters. However, some letters are used more than once. After circling all the words on the list, the unused letters will spell the name of one of the holidays this month. The answer key will follow in next month's BYTE.

G	M	N	E	G	A	T	I	V	E	L	E	V	E	N
X	I	D	A	R	P	O	S	I	T	I	V	E	R	D
N	L	Y	T	A	M	H	T	I	R	A	G	O	L	E
O	L	T	R	T	N	O	Y	T	N	E	W	T	I	R
I	I	F	I	I	E	O	D	E	V	I	F	M	D	D
T	O	I	L	O	E	C	W	U	Y	O	A	U	N	N
C	N	F	L	N	T	N	O	T	L	G	D	E	A	U
A	T	T	I	A	R	N	R	M	I	U	V	I	S	H
R	E	E	O	L	I	I	O	N	P	L	S	G	U	L
F	N	E	N	N	H	T	A	I	E	L	H	H	O	A
S	O	N	I	T	T	R	H	W	L	Y	E	T	H	E
B	I	N	A	R	Y	O	T	R	O	L	T	X	T	R
G	E	X	P	O	N	E	N	T	E	R	I	R	O	U
D	A	D	E	C	I	M	A	L	Y	E	E	B	O	O
N	E	V	E	S	I	N	T	E	G	E	R	Z	R	F

BILLION
BINARY
COMPLEX
DECIMAL
EIGHT
ELEVEN
EXPONENT
FIFTEEN
FIFTY
FIVE
FORTY
FOUR
FRACTION
HUNDRED
IMAGINARY
INTEGER
LOGARITHM
MILLION
MODULUS
NEGATIVE
NINE
ONE
POSITIVE
RADIX
RATIONAL
REAL
ROOT
SEVEN
SIX
TEN
THIRTEEN
THIRTY
THOUSAND
THREE
TRILLION
TWELVE
TWENTY
TWO
ZERO