Microvec

The Other Type of Video Display

Billy Garrett
POB 18806
Greensboro, NC 27419-8806

If you look closely at the video games in an arcade, you will see two main types of video displays. The "other" type of display is called a vector-graphics display (of course, the most common display is the raster type that operates like an ordinary television). The vector display is easily recognized because the displayed image is formed by line segments called vectors. The image resembles a stick figure or an engineering drawing. Because no low-cost vector displays are available commercially, I decided to design and build one for myself.

This article describes how the Microvec unit operates and how to build one. For less than $225, you can have a complete, stand-alone vector-graphics controller that can produce images like those in photos 1, 2, and 3. It will accept commands to draw lines with endpoints anywhere on a 256- by 256-element grid, and it can display any one of four "pages" of video information stored in memory. By connecting the controller to a common oscilloscope, you will have a vector-display unit that can be hooked to any computer. Interfaces specifically designed for use with the S-100 bus and the Radio Shack TRS-80 Model I computer are also described in this article.

Behind the Screens

Billy Garrett began work on the Microvec project as a junior at the University of South Carolina; the Engineering Department there was very helpful, and special thanks goes to Dr. Robert Pettes and Dr. Bill Eccles for their support and encouragement. The South Carolina Honors College, led by Dr. Mould, helped with the manuscript and photos. Due to the aid of Dr. Wierzba and Dr. Walters, work on the project achieved recognition at the IEEE's Region 3 Student Paper Contest, where it took third place. Norman McEntire encouraged the author to send the paper to BYTE in the form of a construction article. The photos in this article are by Howard Rhinehart.

The Arcade Syndrome

Whenever I finally tear myself away from a video-game arcade, I consider myself lucky to have any money left. By the time I arrive home, I'm ready to start designing my own video game. I have come to realize, however, that TRS-80 BASIC is too slow at handling graphics for my purposes; the only way to write an *interactive real-time* game that uses graphics extensively is to use machine language—a tedious process at best. Even worse is the limited resolution (128 by 48 picture elements) that the TRS-80 has to offer. Though other computers with higher resolution are available, I have found that the raster-scan display, as employed by all current home computers, can be difficult to use.

Raster Scan versus Vectors

A raster display is formed one line at a time, from the top of the screen to the bottom (about 525 horizontal lines for a standard TV picture). In a high-resolution raster-scan display, the image is formed by turning on many small dots, called pixels (picture elements). Because so many pixels are in a small area, the eye blurs them into what appears to be a solid form. Many raster-scan displays are available for personal computing use, and the price depends on the type and resolution of the display desired. Even though raster-scan displays are widely used, they are not ideally suited for *all* display applications.

In a vector-display system, an image on the screen is made up of individual *line segments*, rather than dots. A display of this nature is commonly found in an arcade as part of a game. This kind of display is also used to produce schematics, structural outlines of a building, and can even be used for the complicated task of real-time aircraft simulation. Its main advantage over the raster-scan technique is simplicity of use: only two points must be known in order to draw a line; in a raster-scan device, the line is composed of many pixels, each of which must be kept track of individually and turned on or off.

With a vector display, the host processor only needs to send the starting and ending points of a line, which leaves the host processor with much more free time. Rapid animation can be achieved through the use of a vector-graphics display by employing this extra computing time.

The Problems of a Vector-Graphics Display

The major drawbacks to designing a vector display are twofold. The first problem is refreshing the display in sufficiently short time intervals to prevent the vectors from flickering. The solution to this problem is the use of a dedicated microprocessor built on the same board as the vector-display controller to handle the refresh and I/O (input/output) operations. An alternate solution to this problem would have involved the use of discrete integrated circuits or DMA (direct memory access) techniques; this increases the complexity of the circuit greatly, nearly doubling the number of parts required by the dedicated microprocessor approach.

The second problem is that the display used is not like a normal television. Instead, a vector-graphics generator needs an *x*-*y*-type display. Fortunately, a low-grade oscilloscope can be used, but it must have a *z*-axis input that will either blank or intensify the display. (I had an old Heathkit scope that was just sitting idle; it serves as a fine display for the controller described here.) The bandwidth of the scope is not critical. As long as each channel is rated for 100 kHz or more and both channels are DC-coupled, it should work fine.

General Overview

The basic outline of the Microvec unit is shown in the block diagram in figure 1. The microprocessor is responsible for sending coordinates of every vector's endpoints to the D/A (digital-to-analog) converters. Coordinates of the beginning point are

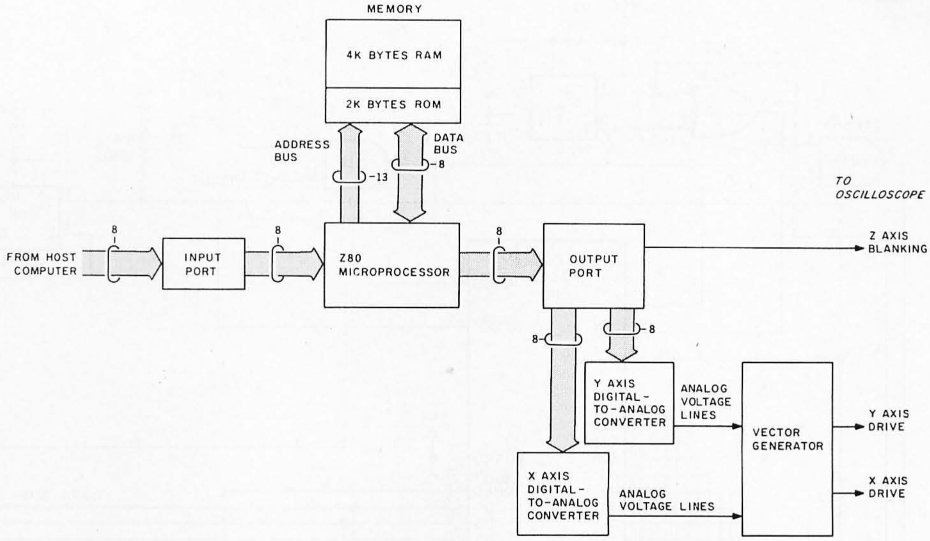


Figure 1: *Block diagram of Microvec. The Z80 microprocessor accepts commands from a host computer system in the form of endpoint coordinates for vectors to be drawn, and transfers them periodically to digital-to-analog converters that drive vector-generating circuitry.*

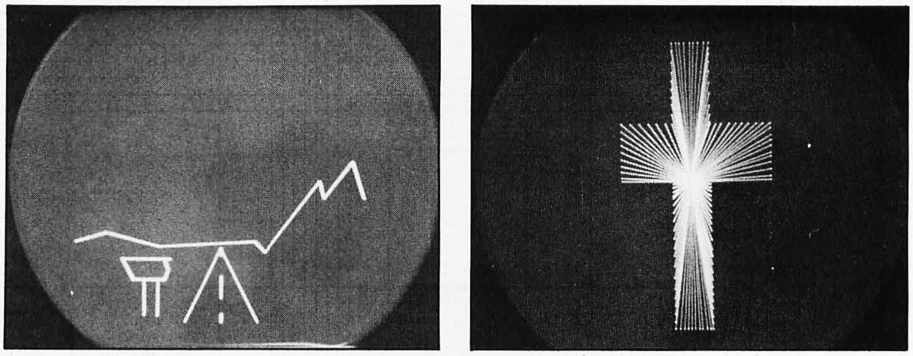


Photo 1: *Images produced by Microvec. These two photos are freehand "doodles" drawn by the author with a TRS-80 Level II BASIC program. The combination of a host computer and the vector controller's dedicated Z80 makes a very flexible and powerful system.*

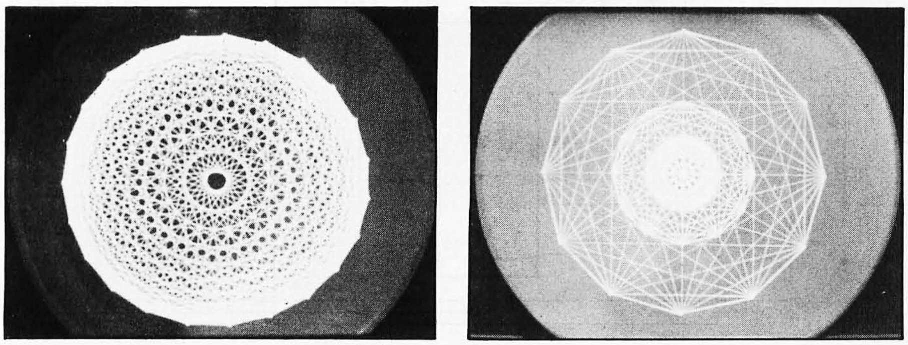


Photo 2: *Circle-forms produced by the LCIRCLE program. These images were designed by simply specifying different parameters before running LCIRCLE. (See listing 2.)*

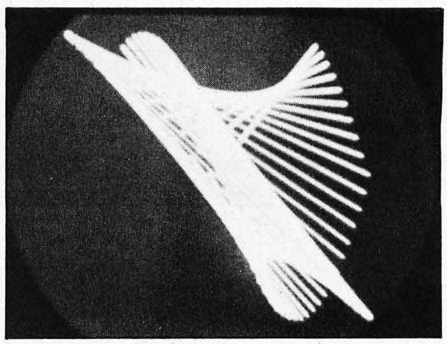


Photo 3: *An image drawn by the STICKS program. This is a vector-scan version of a program commonly used to demonstrate the high-resolution mode of Apple II computers. (See listing 4.)*

(2a)

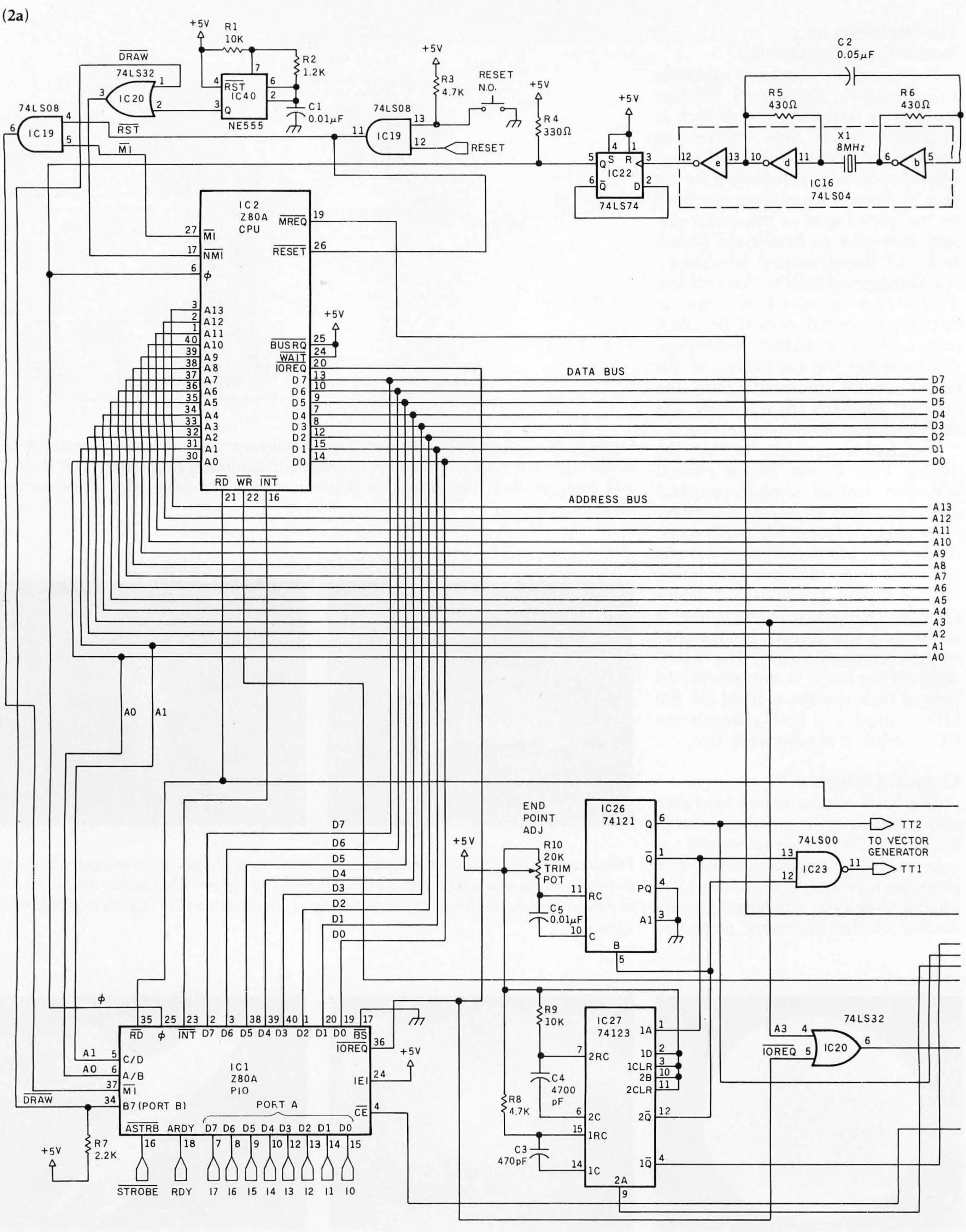


Figure 2: *Schematic diagram of the Microvec system. Figure 2a, the main part of the system, contains the microprocessor, memory, D/A converters (digital-to-analog converters), and vector-generating circuitry. The microprocessor refreshes the display by repeatedly sending all the coordinates in the current memory page to the D/A converters. (Four pages of memory are on board; each can contain 255 vectors.) When the first part of a vector's coordinate is sent to the D/A converters, a voltage is produced that is represen-*

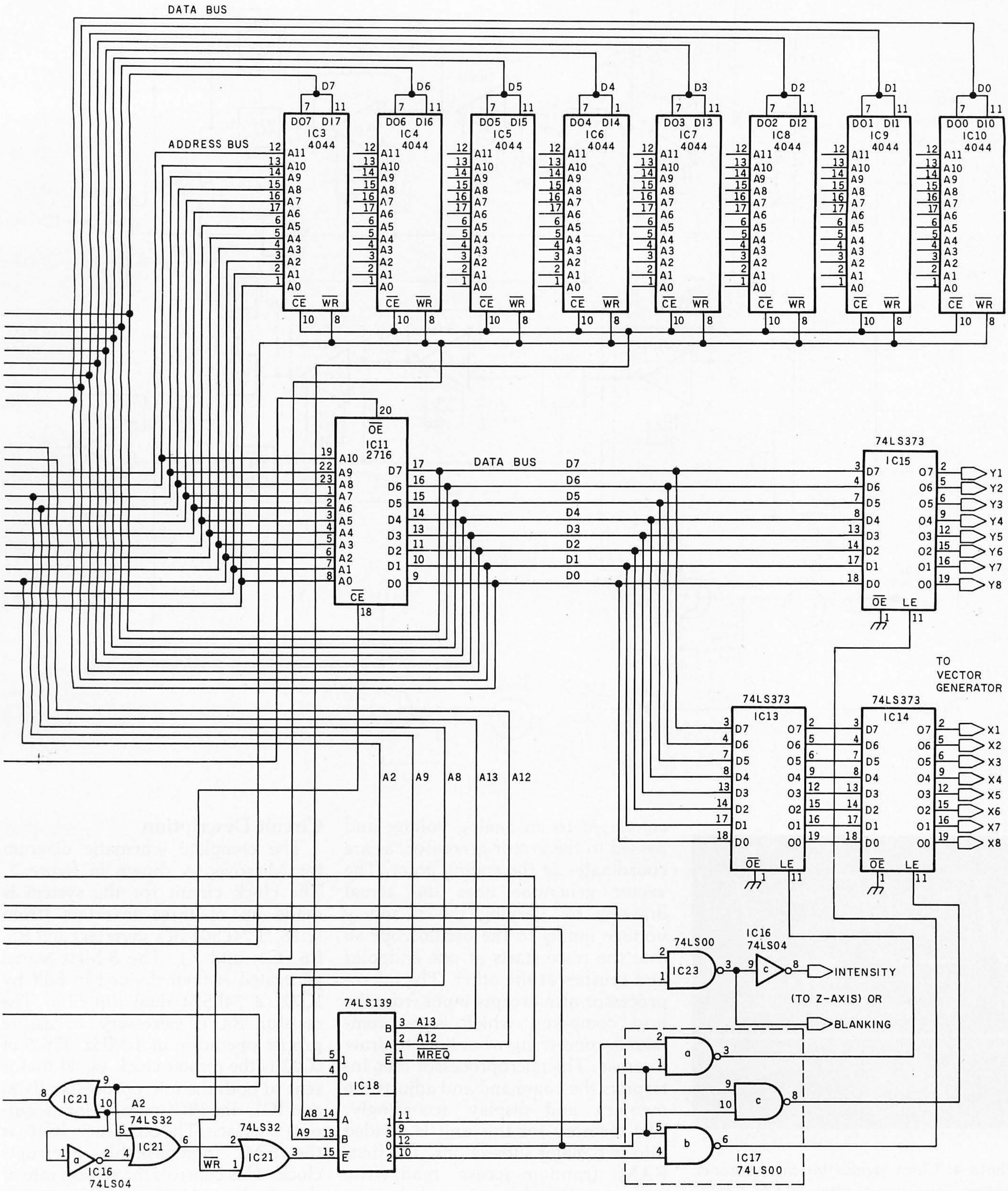
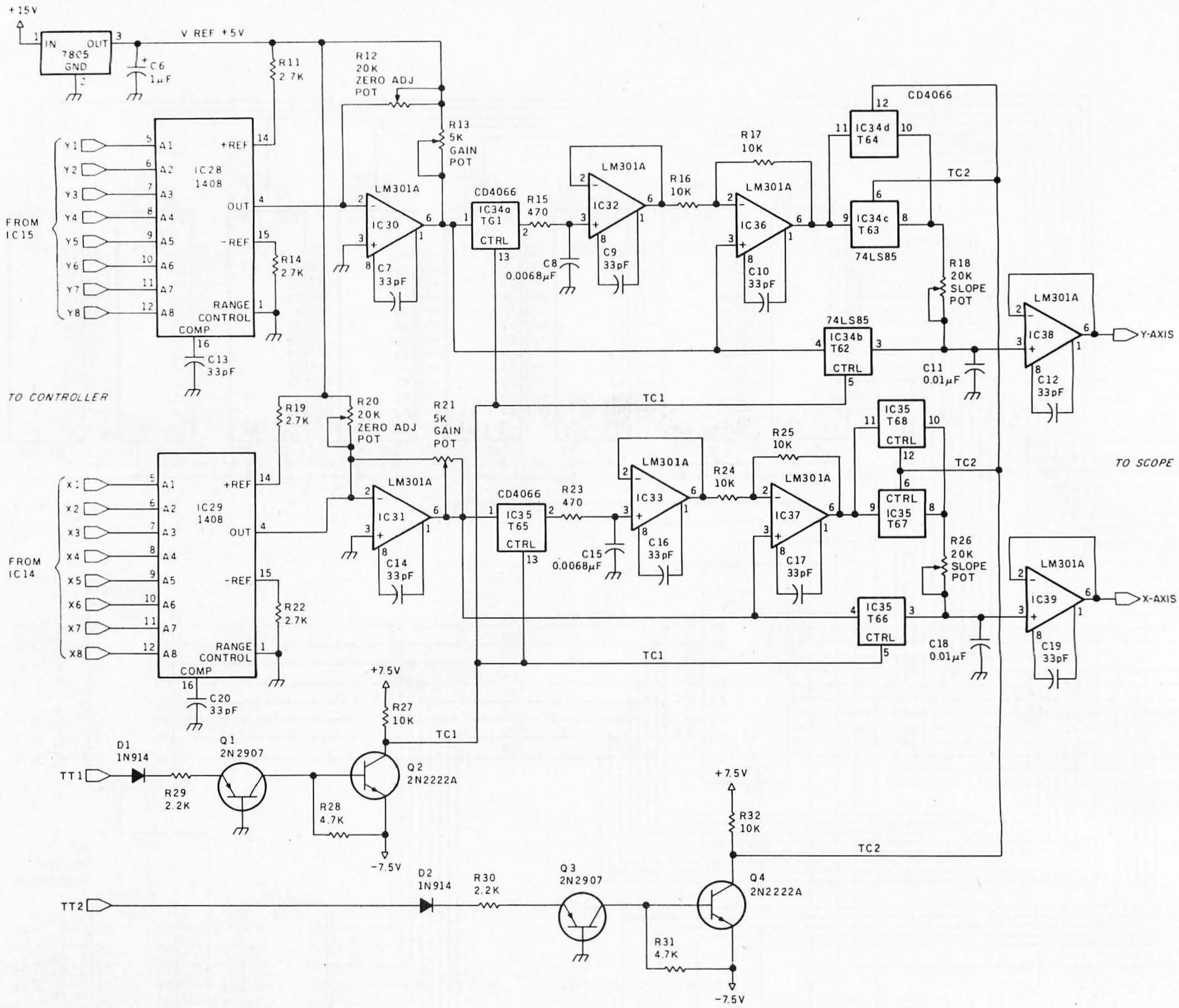


Figure 2a continued on page 512

tative of the starting point of the vector. A series of capacitors is charged to this voltage through a set of CMOS transmission gates. When the second part of the vector's coordinates is presented to the D/A converters, their output voltage becomes representative of the endpoint of the vector, and the voltage on the capacitors changes to this new value. The state of the transmission gates is changed so that the changing voltage on the capacitors is used to drive the display device (an oscilloscope).

(Figure 2a continued)



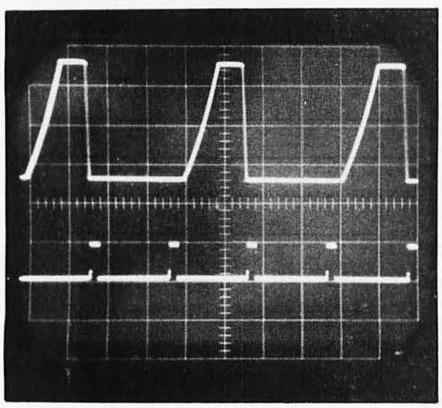


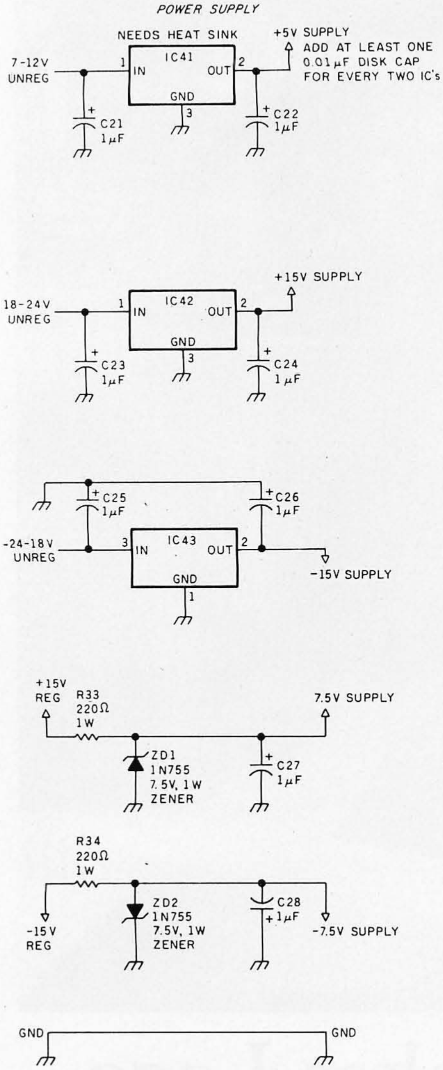
Photo 4: *Clock signals for the Microvec system. The upper trace is the 4-HMz system clock; the lower trace is the adjustable refresh clock. The refresh clock determines how the processor divides its time between refreshing the screen and processing the host computer's commands.*

converted to an analog voltage and passed to the vector generator, as are coordinates of the ending point. The vector generator does the actual drawing by varying the x and y voltage inputs to the oscilloscope so that the trace starts at one endpoint and finishes at the other. The microprocessor also accepts input from the host computer, which issues commands concerning what lines to draw or erase. The microprocessor then interprets the command and adjusts the memory and display accordingly. The memory for this unit is divided into two major subsections. The first, RAM (random-access read/write memory), is a place where the microprocessor stores coordinates for the lines to be drawn. The other type is EPROM (erasable programmable read-only memory), in which the microprocessor's control program resides.

Circuit Description

The complete schematic diagram for Microvec is shown in figure 2. The clock circuit for the system is made up of three inverters (from IC16, a 74LS04 hex inverter) and R5, R6, C2, and X1. The 8-MHz signal generated is then divided in half by IC22, a 74LS74 dual flip/flop. The resistor R4 is necessary to assure proper operation at 4 MHz. Pin 5 of IC22 is the master clock signal that is sent to both the microprocessor (IC2) and IC1, the PIO (parallel input/output) interface. The 555 timer, IC40, is the NMI (nonmaskable interrupt) clock. This controls the refresh rate of the display. At the 4-MHz system clock speed, a good refresh rate is obtained if R1 is 10 kΩ. See photo 4 for a look at this signal. The value of R1 may be decreased, but this will degrade the speed of the I/O operations. (In fact, if R1 is set too low, the

(2b)



Number	Type	Gnd	+5V	+15V	−15V	+7.5V	−7.5V
IC1	Z80A-PIO	11	26				
IC2	Z80A-CPU	29	11				
IC3	4044	9	18				
IC4	4044	9	18				
IC5	4044	9	18				
IC6	4044	9	18				
IC7	4044	9	18				
IC8	4044	9	18				
IC9	4044	9	18				
IC10	4044	9	18				
IC11	2716	12	21,24				
IC12	74LS240	10	20				
IC13	74LS373	10	20				
IC14	74LS373	10	20				
IC15	74LS373	10	20				
IC16	74LS04	7	14				
IC17	74LS00	7	14				
IC18	74LS139	8	16				
IC19	74LS08	7	14				
IC20	74LS32	7	14				
IC21	74LS32	7	14				
IC22	74LS74	7	14				
IC23	74LS00	7	14				
IC24	74LS85	8	16				
IC25	74LS85	8	16				
IC26	74121	7	14				
IC27	74123	8	16				
IC28	1408	2	13		3		
IC29	1408	2	13		3		
IC30	LM301A			7	4		
IC31	LM301A			7	4		
IC32	LM301A			7	4		
IC33	LM301A			7	4		
IC34	CD4066					14	7
IC35	CD4066					14	7
IC36	LM301A			7	4		
IC37	LM301A			7	4		
IC38	LM301A			7	4		
IC39	LM301A			7	4		
IC40	NE555	1	8				

Figure 2b *gives the specifics of power-supply filtering (be sure to see the power-supply connection table accompanying this figure).*

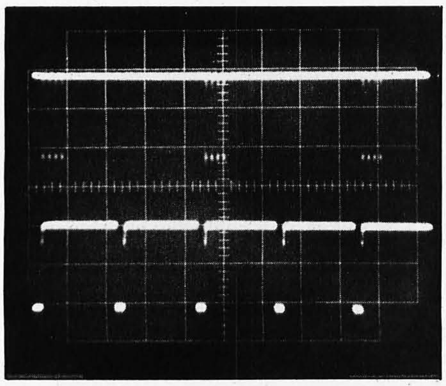


Photo 5: *Timing signals that operate the vector-generating portion of the circuit. The lower trace is the 4-MHz system clock; the upper trace shows the strobe signals sent to the enable pin of IC18, a decoder. The decoder provides four separate, properly timed signals to operate the latches and transmission gates that make up the vector generator.*

microprocessor will spend almost all its time doing video refresh, and the unit will be very sluggish in responding to commands.)

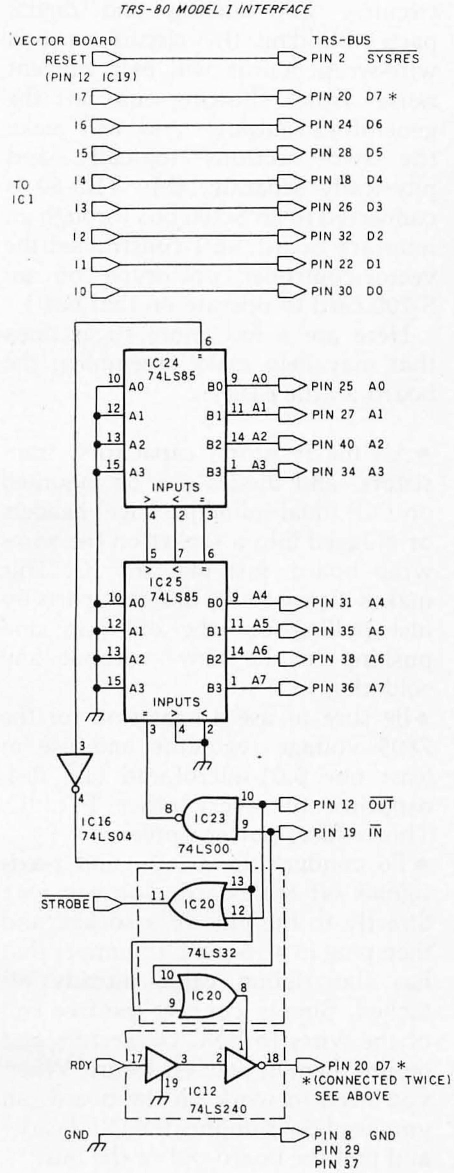
The microprocessor does have some control over the NMI requests. By raising pin 34 of IC1 high, the microprocessor can override the NMIs. This is essential to reset the system. The pins not shown for IC1 are part of "port B" and are not used in this design; therefore, they should be left disconnected. (Any other pins not shown are not used and should also be left alone.)

The memory, IC3 through IC11, is connected to the address bus. Each of these 4044 static memories is connected to *one* data line. Half of IC18, a dual 1-of-4 decoder, is used to select either the ROM (read-only memory) or the RAM.

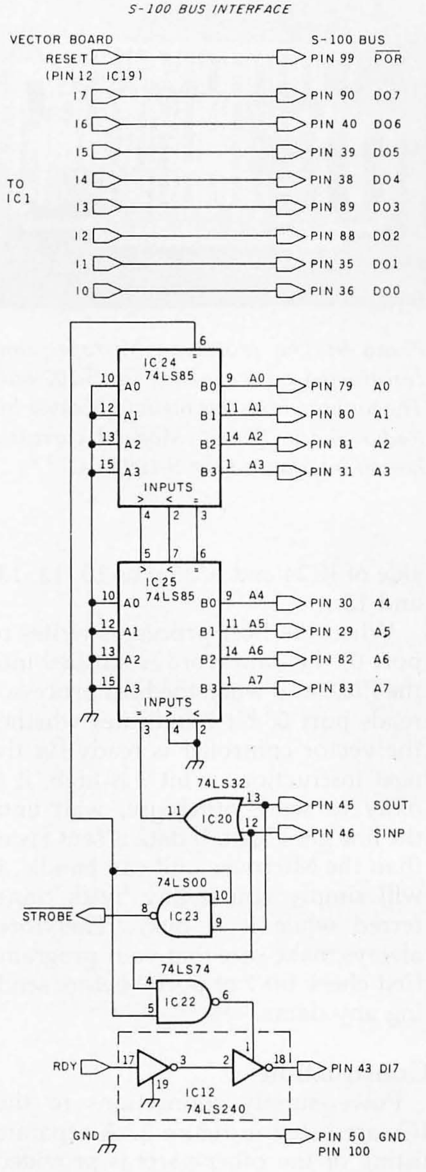
The other half of IC18 decodes the strobes that are used to "fire" the vector generator. See photo 5 for a look at the enable signal on pin 15 of IC18. These four pulses are decoded by IC18 and are used to load and fire the vector generator. First, pin 9 of IC18 will pulse low, followed by pin 10, pin 11, and pin 12, in that order. This causes the proper bytes of data to be loaded into IC13, IC14, and IC15 (all are 74LS373 8-bit latches) and then sent on to the D/A converters. IC26 and IC27 (both are types of one-shots) handle the timing for the vector generator and are triggered by the last strobe, from pin 12 of IC18.

Because the x and y channels of the vector-generator circuitry are built identically, just the y-axis part of the circuit will be described. IC15 is connected to the D/A converter, IC28.

(2c)



(2d)



Figures 2c and 2d *are interfaces to the Radio Shack TRS-80 Model I and the S-100 bus, respectively.*

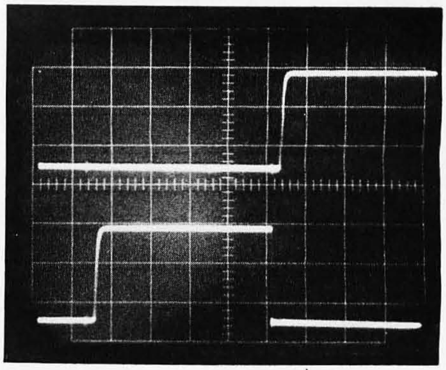


Photo 6: *Translated signals that operate the transmission gates.*

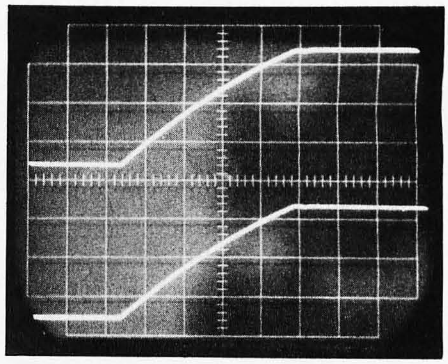


Photo 7: *Output voltages that drive the oscilloscope. The starting and ending points for these "ramps" are determined by the output of the digital-to-analog converters.*

The signal out of pin 4 is a current that is proportional to the input value on pins 5 through 12. The op amp, IC30, converts this current into a voltage that should range over about ±1 volt (V), depending on the gain adjustment, R13. The zero-adjustment potentiometer, R12, is used to set the midpoint of the D/A converter to 0 V. IC34 is a CMOS (complementary metal-oxide semiconductor) 4066 bilateral switch, sometimes called a TG (transmission gate). When its control input (i.e., pin 13) is at 7.5 V, it acts like a switch that has a contact resistance of about 80 ohms; however, when the control pin is at −7.5 V, its resistance is about 1 teraohm (10^{12} Ω), which is a fairly good disconnection! The two signals that control the TGs, TC1 and TC2, are shown in photo 6. Note that these signals are affected by the endpoint-adjustment potentiometer, R10. Therefore, the signals in your circuit will look like that but will not necessarily have exactly the same timing. The four transistors and the parts around them are necessary to translate the TTL-level signals from the one-shots into CMOS levels for use by IC34 and IC35.

Basically, drawing a line occurs through the following sequence of events. First, both D/A converters are given a byte representing the starting point of a line, and their outputs are given a chance to settle. While in this state, TGs 1, 2, 5, and 6 are on, and the other TGs are off. This allows C8, C11, C15, and C18 to charge up to the voltage produced by the corresponding D/A converter. Next, all the TGs are turned off for about 12 microseconds (μs). (Exact timing is determined by IC27, C4, and R9.)

The D/A converters are given the endpoint values and allowed to settle for 12 μs. Then, TGs 3, 4, 7, and 8 are turned on for between 50 and 90 μs (depending upon the endpoint-adjustment setting of R10). During this time, C11 and C18 either charge (if the final value is higher than the initial value) or discharge (if the final value is the lower). The result is that the output voltage takes some time to go from the value stored in the capacitors to the value being generated by the D/A converters. This produces a "ramp" waveform, as shown in photo 7.

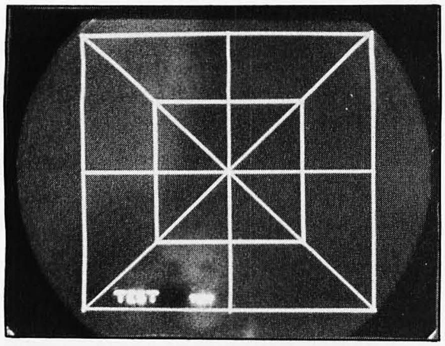


Photo 8: *The built-in test and alignment pattern. Microvec automatically stores these vectors in memory whenever it is turned on, reset, or when the TEST command is given.*

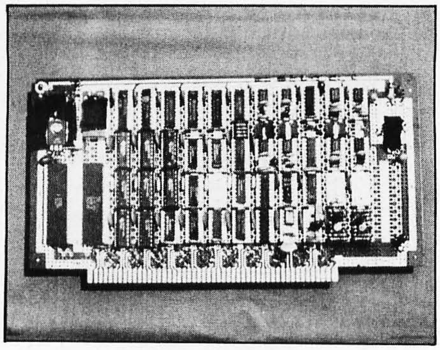


Photo 9: *The prototype Microvec unit constructed on a card for the S-100 bus. The author chose this method because his Radio Shack TRS-80 Model I currently has an interface to the S-100 bus.*

The charge rate is determined by the difference between the stored voltage and the final voltage. IC36 charges C11 through R18, and IC37 charges C18 through R26. Since the time constant (RC) of both sections must be equivalent, be sure C11 and C18 are very closely matched. The last op amp (IC38 and IC39) is used to buffer the output so that the scope will not affect the charging of C11 or C18.

Note in the photo that the outputs are not linear ramps. Instead, the capacitors charge in an exponential fashion (but since they both charge exponentially and have the same time constant, the display seems to produce straight lines). Only the most observant person is going to notice that the lines appear brighter toward the end of the line segment. Much more noticeable is that short vectors are brighter than longer vectors; this is because each vector is "on" for exactly the same length of time. That is why the letters in the words "TEST" in photo 8 are much brighter than the segments of the box. Although this is somewhat annoying, it is easy to get used to.

The power-supply regulators and zener diodes are connected in a normal way, as is shown in figure 2b. The two different host-computer interfaces are also quite simple. The data bus is attached to the "port A" inputs of the PIO, IC1. The lower 8 bits of the address bus are monitored by IC24 and IC25 (see figure 2c and figure 2d); the ICs are set up to enable I/O when the host addresses port 0. (If you want to change this address, simply alter the pattern on the "A" side of IC24 and IC25 pins 10, 12, 13, and 15.)

When the host processor writes to port 0, the data word is strobed into the PIO; and when the host processor reads port 0, bit 7 indicates whether the vector controller is ready for the next instruction. If bit 7 is high, it is okay to send; otherwise, wait until the line goes high. If data is sent faster than the Microvec unit can handle, it will simply ignore any bytes transferred while it is busy. Therefore, always make sure that your programs first check bit 7 of port 0 before sending any data.

Construction

Power-supply connections to the ICs are listed in figure 2. A separate listing of the other parts is provided in the text box "A Shopping List" on page 519. Make a photocopy of the schematic and as you attach each wire, check it off. That way, if you are interrupted or need to leave the board for a while, you will know where you are when you return. You will find that if you are not careful in the building phase, you will spend a great amount of your time tracking down wiring errors; thus, as a word to the wise, go slowly and double-check every wire. Note that the D/A converters have an unusual arrangement of their address pins in that A1 is the *most* significant bit rather than the *least* significant bit. Make sure that you connect the lines as shown.

As you can see from photo 9, the prototype is constructed on an S-100 wire-wrap board. If you do not plan to install your board in an S-100 system, I recommend that you use two separate boards and divide the circuitry into analog and digital parts. Building the circuit on two wire-wrap boards will help prevent noise from showing up in the generator's outputs and will make the two sections logically and physically separate. (My TRS-80 is connected to an S-100 bus through an interface board, so I constructed the vector-controller prototype on an S-100 card to operate on that bus.)

Here are a few more suggestions that may help make assembling the board a little easier:

- All the resistors, capacitors, transistors, and diodes can be mounted on DIP (dual-inline package) headers or plugged into a socket on the wire-wrap board, just like any IC. This makes it easy to try different parts by just pulling out the old part and pushing in the new—without any soldering.
- Be sure to use a heat sink for the 7805 voltage regulator and use at least one 0.01-microfarad (μF) disk capacitor for every other TTL IC. This reduces power ripple.
- To conduct the *x*-, *y*-, and *z*-axis signals off the board, you can wire directly to the pins of a socket, and then plug in a 16-pin DIP jumper that has flat ribbon cable already attached. Simply connect the free end of the wires to BNC connectors and mount them in your enclosure. When you need to work on the board, all you need do is unplug the DIP header and pull the board out of the bus.
- Note that both blanking and intensify signals are provided, although only one will be needed.

Make sure that you buy Z80A *4-MHz-rated* parts. At the time of this writing, no 2716 EPROM is rated for 4 MHz, but a 2716-1 has a 350-nanosecond (ns) access time and should work fine. After I built my board, I substituted parts rated at 2.5 MHz, and they seemed to work fine at the 4-MHz rate; but there is no guarantee that this will be the case for all 2.5-MHz parts over even a narrow temperature range.

Software

The program I have written to drive Microvec's Z80 divides the available memory into 255-vector "pages." This allows the user to display or work on any of the four pages independently. In this arrangement, the available memory is almost completely used. The memory for storing vectors could easily be expanded to 62 pages; however, that many extra pages would force the design to occupy several boards. Any one of the four different pages can be displayed while the user is editing that page (or any other page). This leaves at least two pages for storage.

Editing functions include commands for adding a new vector, deleting a given vector, clearing the entire page, deleting the last vector, and much more. See table 1 for a summary of the commands.

Figure 3 is a representation of the points available on the screen. If you want to draw a line across the bottom of the screen, you would need to send the following sequence of bytes: 4,0,0,255,0 (the first byte is the command—in this case, draw a line; the following 4 bytes give the starting point [0,0] and the ending point [255,0]). To erase that line, you could send the command 5,0,0,255,0; or you could simply send the command to delete the last vector: 6. This would have turned off the one line that was lit.

The commands DRAW, UNDRAW, CLEAR, and DELETE affect the MPAGE, which is the page you are editing. Use the command MODP to change the page number you want to work in. Initially, MPAGE is set to page 0. Use the command DISP to select DPAGE, the page you want displayed. DPAGE is initially set to 0. If you draw a line on one page and are displaying another, the displayed image will not change. The only way to watch the unit add or delete vectors is to set MPAGE and DPAGE to the same number.

The command TEST is the only command, other than RST, that will affect the display even if MPAGE and DPAGE are not the same. It produces the image in photo 8 by putting the vectors in memory; it does not reset the unit. The RST command does, however, reset the unit. Whenever Microvec is turned on, or reset to its initial state, it automatically performs a TEST command. Thus, the effect of RST is to both reset and test the unit. It takes a special command sequence to accomplish the RST function: 0,0,0,0,7,15. The "15" is a simple form of insurance; should a "7" code be sent accidentally, nothing will

A Shopping List

As a help, here is a condensed shopping list of necessary parts. The prices shown in this list are approximate and representative of parts available through mail-order ads in various issues of BYTE. Since this article was written, the prices will almost certainly have gone down on some of the parts.

The only hard-to-get part is the programmed EPROM. If you are lucky enough to have an EPROM programmer, you can "burn" the program into the EPROM yourself. If you want a 2716 programmed, there is a charge of $10 to copy the program on each 2716 (no other type EPROM), which must be erased. You will receive the latest version of the program. However, there is no guarantee expressed or implied. Additionally, a disk is available with all the programs in this article, plus several other programs and the source code *to the EPROM, for $15. Send a check or money order (including $2 for postage and handling) to Garco, POB 18806, Greensboro, NC 27419-8806. If you order a copy of either the EPROM or the disk, you'll also receive any errata sheets and additional documentation. Because I'm also planning on providing a revised EPROM, I need to know how many people would be interested. Those that send in an order will be on a mailing list. This offer is subject to change at any time.*

Quantity	Part	Amount
1	Z80A-CPU	$10.50
1	Z80A-PIO	8.60
8	4044 RAMs	31.92
1	2716-1 Preprogrammed	26.00
1	74LS240 Octal TS Inverter	1.65
3	74LS373 Octal Latches	4.35
1	74LS04 Hex Inverter	0.25
2	74LS00 Quad Nand Gates	0.50
1	74LS139 Dual 2-4 Decoder	0.75
1	74LS08 Quad And Gate	0.35
2	74LS32 Quad Or Gates	0.70
1	74LS74 Dual Flip/Flop	0.45
2	74LS85 4-Bit Comparators	2.30
1	74121 One-Shot	0.29
1	74123 Dual One-Shot	0.55
2	MC1408L8 8-Bit D/A Converters	11.90
8	LM301A Op Amps	3.12
2	4066 Quad Bilateral Switches	1.50
1	555 Timer	0.39
1	7805 5-Volt Regulator	0.89
1	78L05 Low-Power 5-Volt Regulator	0.79
1	7815 15-Volt Regulator	0.99
1	7915 −15-Volt Regulator	1.19
25	16-Pin WW Gold Sockets	17.25
4	20-Pin WW Gold Sockets	4.39
8	18-Pin WW Gold Sockets	7.92
2	40-Pin WW Gold Sockets	3.98
1	24-Pin WW Gold Socket	1.45
1	Vector WW Board 8800V DP	22.20
1	Reset Switch	1.50
1	8-MHz Crystal	1.99
27	Assorted Resistors	2.70
7	10-Turn Potentiometers	14.00
28	Assorted Capacitors	10.00
4	Transistors	2.00
2	1N914 Diodes	1.00
2	7.5-V 1-W Zener Diodes	1.00
1	16-Pin Jumper Cable (18-inch)	5.40
20	Despiking Caps (0.1 μF)	5.30
1	Heat Sink	0.75
1	Wire-Wrap Wire	9.95
	Total:	**$222.71**

Code	Number of Bytes	Name	Description
1	2	MODP	Modify page (MPAGE)
2	2	DISP	Display page (DPAGE)
3	1	CLEAR	Clear MPAGE
4	5	DRAW	Draw line in MPAGE
5	5	UNDRAW	Erase line in MPAGE
6	1	DEL	Delete last vector in MPAGE
7	2	RST	Reset if key is correct
8	1	TEST	Turn on test pattern

Command Summary

Table 1: *Summary of the commands to Microvec. The host system uses these commands to control the display of the four memory pages. In the table, MPAGE is the page being modified, while DPAGE is the page being displayed.*

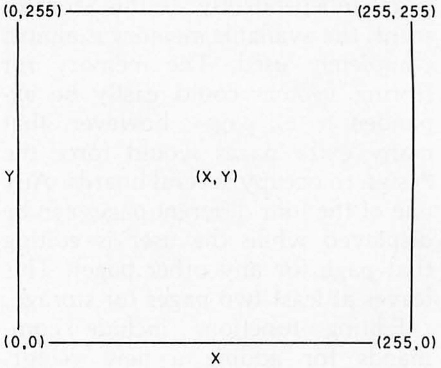


Figure 3: *Representation of the oscilloscope display screen. Microvec can draw vectors with endpoints at any of the coordinates shown.*

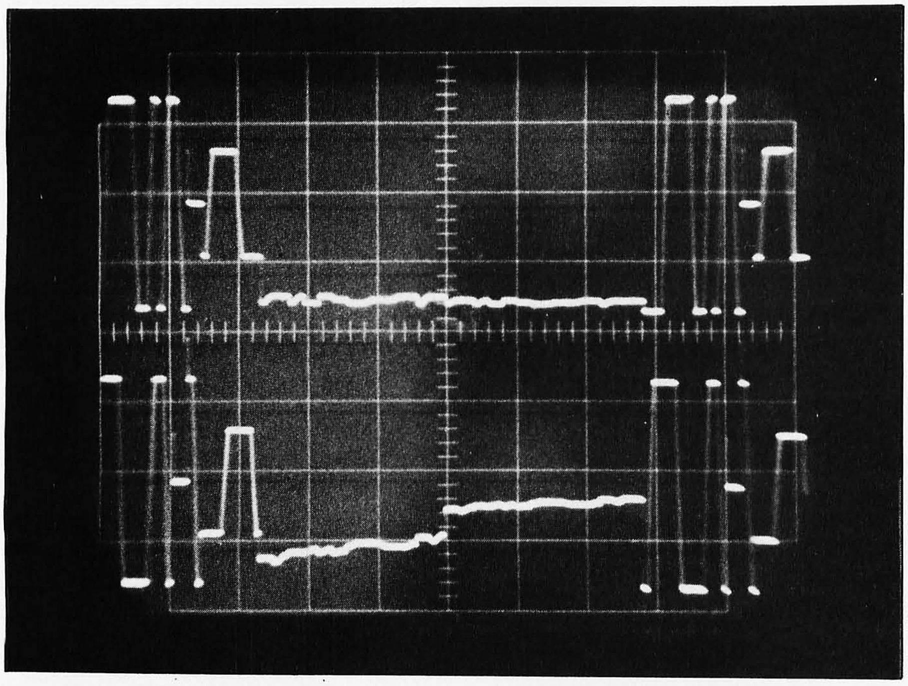


Photo 10: *Comparison of the x- and y-axis driving signals while Microvec is displaying the test pattern of photo 8.*

Listing 1: *Controlling software for Microvec's Z80 processor. This assembly-language program is stored in an EPROM installed on the Microvec circuit card; it allows the Z80 microprocessor to display vectors stored in memory and to accept commands from a host computer system.*

```
                ;*****************************************************
                ;*****************************************************
                ;***** Copyright (C) 1981 By Billy Garrett ***************
                ;*************** MICROVEC V1.0 ***************************
                ;*****************************************************
                ;*****************************************************
                ;
                TITLE Vector Graphics Generator Program   MICROVEC V1.0
0000'                   ASEG            ;Force the assembly at location 0 H
                        ORG     0000H   ;Start
                                        ;Startup routine
                ;
                .COMMENT ^
                This is a model of the registers inside of the Z80 and
                how they are used by this program.  There are some registers
                that have dedicated functions, and their contents must not
```

Listing 1 continued on page 521

happen to all your hard work. The "15" is just a key.

Almost any command you can imagine might be added to the Z80 controller program simply by altering the code in the EPROM. Currently, the program occupies about 800 bytes of the 2048 bytes available. This leaves plenty of room. The assembled program is shown in listing 1.

The program is highly dependent on the interrupt structure of the Z80. Simply stated, two types of interrupts are available to Z80 programmers: the highest priority one is the NMI. Whenever the NMI signal is active, and as soon as the processor finishes the current instruction, the interrupt is acknowledged. The address 66 hexadecimal is placed in the program counter; the program at this location causes the system to draw a line. The important point is that the processor is being interrupted about 10,000 times each second, so that even when the host system is not sending commands to the board, half the Z80's time is being used to redraw the display.

The second type of interrupt is called simply INT. It is used when the host system has information to transfer to Microvec. Because this signal can be "masked" by the Z80, the processor can be programmed to ignore or postpone it.

Calibration

Before installing any ICs, check the output of each voltage regulator. Make sure that their outputs are within 5 percent of their rated values. Then, turn the power off and adjust R12 and R20 to about 5 kΩ. Next, adjust R13 and R21 to about 2 kΩ. After

Text continued on page 527

Listing 1 continued:

```
                       be destroyed.  Also, there are only four (4) levels of stack
                       room available in this 4K byte version, so modifications must
                       not under any circumstances push anything more than 3 deep
                       onto the stack. This is because the NMI can occur at any time
                       and it requires one level of stack.

                          Main Register Set              Alternate Register Set
                       ---------------------------------------------------------
                       | A          | F          | A'           | F'          |
                       |     *      |    Flags   | Vector Count |    Flags    |
                       ---------------------------------------------------------
                       | B          | C          | B'           | C'          |
                       |   DPAGE    |    MPAGE   |  Byte Count  |   Port #    |
                       ---------------------------------------------------------
                       | D          | E          | D'           | E'          |
                       |     *      |     *      |    OOH       |      ?      |
                       ---------------------------------------------------------
                       | H          | L          | H'           | L'          |
                       |     *      |     *      | High .   Pointer  '   Low  |
                       ---------------------------------------------------------
                       | I          | R          |
                       |   O7H      |     ?      |     *= General Purpose
                       ---------------------------          Registers
                       | IX                      |
                       |   MPAGE Base Address    |
                       ---------------------------
                       | IY                      |
                       |   MPAGE 1st Free Space  |
                       ---------------------------
0000  F3               INIT:   DI                  ;Disable interrupts
0001  31 2000                  LD      SP,2000H    ;Set stack pointer to top of memory
0004  AF                       XOR     A           ;Zero A
0005  32 1000                  LD      (NUMVO),A   ;Initialize memory locations
0008  32 1001                  LD      (NUMV1),A   ;to zero
000B  32 1002                  LD      (NUMV2),A   ;
000E  32 1003                  LD      (NUMV3),A   ;
                       ;
0011  21 0244                  LD      HL,MTEST    ;Load HL with the MTEST location
0014  22 1005                  LD      (MEM),HL    ;Store MTEST in page pointer
0017  3E 26                    LD      A,026H      ;Number of vectors in MTEST
0019  32 1004                  LD      (NUMVEC),A;stored in NUMVEC
                       ;
001C  06 FO                    LD      B,OFOH      ;Initialize the registers
001E  0E 05                    LD      C,05H       ;Output port number
0020  16 00                    LD      D,OOH       ;Constant in D
0022  B2                       OR      D           ;Set zero flag (if necessary)
0023  D9                       EXX                 ;Exchange register sets
0024  08                       EX      AF,AF'      ;Including AF
                       ;
0025  AF                       XOR     A           ;Clear A
0026  4F                       LD      C,A         ;Load MPAGE to page 0
0027  47                       LD      B,A         ;Load DPAGE to pageO
0028  DD 21 1007               LD      IX,PAGEO    ;Load modify page address in IX
002C  FD 21 1007               LD      IY,PAGEO    ;Load next free location in IY
                       ;
0030  3E 07                    LD      A,07H       ;
0032  ED 47                    LD      I,A         ;Set the interrupt high byte to 07
0034  ED 5E                    IM      2           ;Set maskable interrupt to mode 2
                       ;
0036  3E 4F                    LD      A,4FH       ;Set the PIO for channel A input
0038  D3 02                    OUT     (02H),A     ;Send control word to the PIO
003A  3E 87                    LD      A,87H       ;Set the PIO to generate interrupts
003C  D3 02                    OUT     (02H),A     ;Send it to the PIO
003E  21 07FE                  LD      HL,INRPTV   ;Load the address of the interrupt
0041  7D                       LD      A,L         ;vector into A
0042  D3 02                    OUT     (02H),A     ;Send it to the PIO
0044  DB 00                    IN      A,(OOH)     ;Dummy read to set the READY line
0046  3E OF                    LD      A,OFH       ;Instruction for the PIO
0048  D3 03                    OUT     (03H),A     ;port B to be output
004A  3E 00                    LD      A,OOH       ;Send to port B
004C  D3 01                    OUT     (01H),A     ;Sets bit seven low on port B, and
                                                   ;Starts the CPU accepting NMI'S
004E  18 3E                    JR      FIRST       ;Now go to the main input routine
```

Draw Interrupt (NMI)

```
                               SUBTTL Draw Interrupt (NMI)
                       ;This routine uses the alternate register set and assumes
                       ;that the main program does not alter it.
                       ;The contents of the alternate register set are defined as:
                       ; A=Number of vectors left to draw in that page
                       ; B=Vector counter (when the two low bits of this register
                       ; are decoded, they are used to fire the vector generator
                       ; C=OUTPUT PORT=05H
                       ; D=OOH, a constant used to speed up the process
                       ; HL=Pointer to vector in memory
                       ;
                               ORG     066H        ;Location of non-maskable interrupt
0066  08               INRPT:  EX      AF,AF'      ;Get alternate register set
0067  D9                       EXX                 ;Including BC,DE,HL
0068  28 19                    JR      Z,NOMORE    ;If zero jump to NOMORE
006A  ED A3                    OUTI                ;Output four bytes
006C  00                       NOP                 ;NOPs necessary for delay
006D  00                       NOP                 ;so that the D/A converters
006E  00                       NOP                 ;have time to settle
006F  00                       NOP                 ;
0070  ED A3                    OUTI                ;
0072  00                       NOP                 ;
0073  00                       NOP                 ;
0074  00                       NOP                 ;
0075  00                       NOP                 ;
0076  ED A3                    OUTI                ;
0078  00                       NOP                 ;
0079  00                       NOP                 ;
007A  00                       NOP                 ;
007B  00                       NOP                 ;
007C  ED A3                    OUTI                ;
007E  3D                       DEC     A           ;Decrement the number of vectors
007F  D9                       EXX                 ;Save the status of the registers
0080  08                       EX      AF,AF'      ;including AF
0081  ED 45                    RETN                ;Return from non-maskable interrupt
0083  3A 1004          NOMORE: LD      A,(NUMVEC);Reload A with the # of vectors
0086  B2                       OR      D           ;Set the zero flag if zero
0087  2A 1005                  LD      HL,(MEM)    ;Reset HL to the start of the page
008A  D9                       EXX                 ;Save status
008B  08                       EX      AF,AF'      ;
008C  ED 45                    RETN                ;Return
```

Inwait Macro

```
                               SUBTTL Inwait Macro
                       INWAIT  MACRO               ;Define macro INWAIT
                               LOCAL WAITON        ;Make waiton have different labels
                               SCF                 ;Set carry flag
                               EI                  ;Enable interrupts
                       WAITON: JR      C,WAITON    ;Wait on interrupt
                               ENDM                ;End MACRO
```

Main Routine

```
                               SUBTTL Main Routine
                       FIRST:  INWAIT              ;Wait on first input
008E  37                       SCF                 ;Set carry flag
008F  FB                       EI                  ;Enable interrupts
0090  38 FE            ..0000: JR      C,..0000    ;Wait on interrupt
0092  21 1007                  LD      HL,PAGEO    ;Now initialize the values
0095  F5                       PUSH    AF          ;Save AF for a moment
0096  AF                       XOR     A           ;Clear A
0097  32 1004                  LD      (NUMVEC),A;Zero the displayed values
009A  22 1005                  LD      (MEM),HL    ;Correct the memory start address
```

Listing 1 continued on page 522

Listing 1 continued:

```
        009D    F1                  POP     AF          ;Restore AF
        009E    C3 00A5             JP      SKIP        ;Skip first inwait, already done
                            ;
                            ;
                            INPUT:  INWAIT              ;Wait on input
        00A1    37                  SCF                 ;Set carry flag
        00A2    FB                  EI                  ;Enable interrupts
        00A3    38 FE       ..0001: JR      C,..0001    ;Wait on interrupt
        00A5    11 0708     SKIP:   LD      DE,JTABLE   ;Load DE with jtable base address
        00A8    E6 7F               AND     7FH         ;Make sure the value is positive
        00AA    FE 09               CP      NUMCOM      ;Is the command greater than NUMCOM?
        00AC    F2 00A1             JP      P,INPUT     ;If out of range, go back to input
        00AF    87                  ADD     A,A         ;Shift A right
        00B0    83                  ADD     A,E         ;Add the offset
        00B1    5F                  LD      E,A         ;Put it back
        00B2    1A                  LD      A,(DE)      ;Load A indirect
        00B3    6F                  LD      L,A         ;Put low byte in L
        00B4    13                  INC     DE          ;Point to next byte
        00B5    1A                  LD      A,(DE)      ;Load A indirect again
        00B6    67                  LD      H,A         ;Put the high byte in H
        00B7    E9                  JP      (HL)        ;Jump indirect
```

Modify Page Command

```
                                    SUBTTL Modify Page Command
                            MODP:   INWAIT              ;Perform inwait macro
        00B8    37                  SCF                 ;Set carry flag
        00B9    FB                  EI                  ;Enable interrupts
        00BA    38 FE       ..0002: JR      C,..0002    ;Wait on interrupt
        00BC    E6 03               AND     03H         ;Mask off the top 5 bits
        00BE    4F                  LD      C,A         ;Store new page number
        00BF    87                  ADD     A,A         ;Multiply A by two
        00C0    21 0700             LD      HL,TABLE    ;Set HL pointer to table
        00C3    85                  ADD     A,L         ;
        00C4    6F                  LD      L,A         ;
        00C5    5E                  LD      E,(HL)      ;Load A indirect
        00C6    23                  INC     HL          ;Point to next byte
        00C7    56                  LD      D,(HL)      ;Load indirect
        00C8    D5                  PUSH    DE          ;Push the start address on stack
        00C9    DD E1               POP     IX          ;And put in IX
                            ;
        00CB    21 1000             LD      HL,NUMVO    ;Load pointer with base address
        00CE    79                  LD      A,C         ;Get MPAGE in A
        00CF    85                  ADD     A,L         ;
        00D0    6F                  LD      L,A         ;
        00D1    6E                  LD      L,(HL)      ;Get the # of vectors in that page
        00D2    26 00               LD      H,00H       ;Clear H
        00D4    29                  ADD     HL,HL       ;Multiply the # of vectors by four
        00D5    29                  ADD     HL,HL       ;
        00D6    19                  ADD     HL,DE       ;Find end of mem and save in HL
        00D7    E5                  PUSH    HL          ;put HL on stack
        00D8    FD E1               POP     IY          ;put next free byte of MPAGE in IY
        00DA    C3 00A1             JP      INPUT       ;Go back
```

Display Page Command

```
                                    SUBTTL Display Page Command
                            DISP:   INWAIT              ;Wait on character
        00DD    37                  SCF                 ;Set carry flag
        00DE    FB                  EI                  ;Enable interrupts
        00DF    38 FE       ..0003: JR      C,..0003    ;Wait on interrupt
        00E1    E6 03               AND     03H         ;Mask off top 5 bits
        00E3    47                  LD      B,A         ;Load DPAGE into B
        00E4    87                  ADD     A,A         ;Multiply by two
        00E5    21 0700             LD      HL,TABLE    ;Load HL with table address
        00E8    85                  ADD     A,L         ;
        00E9    6F                  LD      L,A         ;
        00EA    5E                  LD      E,(HL)      ;Load E indirect
        00EB    23                  INC     HL          ;Point to next byte
        00EC    56                  LD      D,(HL)      ;Load D indirect
        00ED    21 1000             LD      HL,NUMVO    ;Load HL with pointer
        00F0    78                  LD      A,B         ;Get DPAGE number in A
        00F1    85                  ADD     A,L         ;
        00F2    6F                  LD      L,A         ;
        00F3    66                  LD      H,(HL)      ;Load H indirect
        00F4    DB 01               IN      A,(01H)     ;Get the value in port B
        00F6    F6 80               OR      080H        ;Set bit seven high
        00F8    D3 01               OUT     (01H),A     ;Send it back to turn display off
        00FA    7C                  LD      A,H         ;Restore A
        00FB    32 1004             LD      (NUMVEC),A;Save the # of vectors in NUMVEC
        00FE    ED 53 1005          LD      (MEM),DE    ;Load page pointer to mem location
        0102    DB 01               IN      A,(01H)     ;Turn the display back on
        0104    E6 7F               AND     07FH        ;
        0106    D3 01               OUT     (01H),A     ;
        0108    C3 00A1             JP      INPUT       ;Go back
```

Clear Current MPAGE

```
                                    SUBTTL Clear Current MPAGE
        010B    79          CLEAR:  LD      A,C         ;Load A with the # of the MPAGE
        010C    21 1000             LD      HL,NUMVO    ;Pointer to bottom of table
        010F    85                  ADD     A,L         ;
        0110    6F                  LD      L,A         ;
        0111    AF                  XOR     A           ;Clear A
        0112    77                  LD      (HL),A      ;Zero that page's count
        0113    78                  LD      A,B         ;Get DPAGE in A
        0114    B9                  CP      C           ;Is it the same as MPAGE?
        0115    20 04               JR      NZ,OK       ;If not then OK
        0117    AF                  XOR     A           ;Otherwise clear A
        0118    32 1004             LD      (NUMVEC),A;Zero the NUMVEC location
        011B    DD E5       OK:     PUSH    IX          ;Put Bottom of MPAGE addr on stack
        011D    FD E1               POP     IY          ;That's the next free address
        011F    C3 00A1             JP      INPUT       ;Go back
```

Draw New Vector Command

```
                                    SUBTTL Draw New Vector Command
        0122    79          DRAWV:  LD      A,C         ;Load MPAGE in A
        0123    21 1000             LD      HL,NUMVO    ;Load base address in HL
        0126    85                  ADD     A,L         ;Add index
        0127    6F                  LD      L,A         ;
        0128    7E                  LD      A,(HL)      ;Load A indirect
        0129    3C                  INC     A           ;Add one to the number of vectors
        012A    C2 0140             JP      NZ,HERE     ;If not zero then ok
                                    INWAIT              ;Otherwise, ignore the next
        012D    37                  SCF                 ;Set carry flag
        012E    FB                  EI                  ;Enable interrupts
        012F    38 FE       ..0004: JR      C,..0004    ;Wait on interrupt
                                    INWAIT              ;four inputs
        0131    37                  SCF                 ;Set carry flag
        0132    FB                  EI                  ;Enable interrupts
        0133    38 FE       ..0005: JR      C,..0005    ;Wait on interrupt
                                    INWAIT              ;
        0135    37                  SCF                 ;Set carry flag
        0136    FB                  EI                  ;Enable interrupts
        0137    38 FE       ..0006: JR      C,..0006    ;Wait on interrupt
                                    INWAIT              ;
        0139    37                  SCF                 ;Set carry flag
        013A    FB                  EI                  ;Enable interrupts
        013B    38 FE       ..0007: JR      C,..0007    ;Wait on interrupt
        013D    C3 00A1             JP      INPUT       ;Go back
                            ;
        0140    57          HERE:   LD      D,A         ;Keep new number in D
        0141    1E 04               LD      E,04H       ;Set up a counter for four inputs
                            ;
                            LOOP:   INWAIT              ;Wait for input
        0143    37                  SCF                 ;Set carry flag
        0144    FB                  EI                  ;Enable interrupts
        0145    38 FE       ..0008: JR      C,..0008    ;Wait on interrupt
        0147    FD 77 00            LD      (IY+0),A    ;Save vector in next free space
```

Listing 1 continued:

```
014A  FD 23                 INC   IY          ;point to the next free space
014C  1D                    DEC   E           ;Decrement count
014D  20 F4                 JR    NZ,LOOP     ;Loop back
014F  72                    LD    (HL),D      ;Now, increase the number of vectors
0150  78                    LD    A,B         ;Load DPAGE in A
0151  B9                    CP    C           ;Is it MPAGE?
0152  C2 00A1               JP    NZ,INPUT    ;If not, then done
0155  7A                    LD    A,D         ;Otherwise, put the number in A
0156  32 1004               LD    (NUMVEC),A;Store in NUMVEC
0159  C3 00A1               JP    INPUT       ;Go back
```

Undraw Vector Command

```
                            SUBTTL Undraw Vector Command
015C  21 1000     UNDRAW:   LD    HL,NUMVO    ;Load HL with a base pointer
015F  79                    LD    A,C         ;Get the MPAGE number in A
0160  85                    ADD   A,L         ;
0161  6F                    LD    L,A         ;
0162  7E                    LD    A,(HL)      ;Get the # of vectors in the MPAGE
0163  F5                    PUSH  AF          ;Save it
                            INWAIT            ;Wait on input
0164  37                    SCF               ;Set carry flag
0165  FB                    EI                ;Enable interrupts
0166  38 FE       ..0009:   JR    C,..0009    ;Wait on interrupt
0168  57                    LD    D,A         ;Put X1 in D
                            INWAIT            ;Wait on input
0169  37                    SCF               ;Set carry flag
016A  FB                    EI                ;Enable interrupts
016B  38 FE       ..000A:   JR    C,..000A    ;Wait on interrupt
016D  5F                    LD    E,A         ;Put Y1 in E
                            INWAIT            ;Wait on input
016E  37                    SCF               ;Set carry flag
016F  FB                    EI                ;Enable interrupts
0170  38 FE       ..000B:   JR    C,..000B    ;Wait on interrupt
0172  67                    LD    H,A         ;Put X2 in H
                            INWAIT            ;Wait on input
0173  37                    SCF               ;Set carry flag
0174  FB                    EI                ;Enable interrupts
0175  38 FE       ..000C:   JR    C,..000C    ;Wait on interrupt
0177  6F                    LD    L,A         ;Put Y2 in L
0178  F1                    POP   AF          ;Get AF back
0179  F6 00                 OR    00H         ;Check for any vectors in page
017B  CA 00A1               JP    Z,INPUT     ;If not, go back
017E  DD E5                 PUSH  IX          ;Otherwise, save IX for later
                  ;
0180  F5                    NOFIND: PUSH AF         ;Save AF
0181  DD 7E 00                      LD   A,(IX+0)   ;Load A with first entry
0184  BA                            CP   D          ;Is it the same as X1?
0185  28 0E                         JR   Z,FIND1    ;If so, jump to FIND1
0187  DD 23                 WELL:   INC  IX         ;Increnent IX by four
0189  DD 23                         INC  IX         ;
018B  DD 23                         INC  IX         ;
018D  DD 23                         INC  IX         ;
018F  F1                            POP  AF         ;Get AF back
0190  3D                            DEC  A          ;Decrement the counter
0191  20 ED                         JR   NZ,NOFIND  ;Loop
0193  18 55                 JR    ADONE       ;Go back if not found
                  ;
0195  DD 7E 01    FIND1:    LD    A,(IX+1)    ;Is the next entry Y1?
0198  BB                    CP    E           ;Find out!
0199  20 EC                 JR    NZ,WELL     ;If not, jump to WELL
019B  DD 7E 02              LD    A,(IX+2)    ;Load A with the next entry
019E  BC                    CP    H           ;Is it X2?
019F  20 E6                 JR    NZ,WELL     ;If not, jump to WELL
01A1  DD 7E 03              LD    A,(IX+3)    ;Load A with the next entry
01A4  BD                    CP    L           ;Could it be!!!?
01A5  20 E0                 JR    NZ,WELL     ;If not, jump to well
01A7  F1                    POP   AF          ;Otherwise, don't need AF any more
01A8  FD E5                 PUSH  IY          ;Put IY on the stack
01AA  E1                    POP   HL          ;Put the end of the table into HL
01AB  DD E5                 PUSH  IX          ;Put the found address on the stack
```

Undraw Vector Command

```
01AD  D1                    POP   DE          ;Put it in DE
01AE  37                    SCF               ;Set the carry flag
01AF  3F                    CCF               ;Complement the carry flag
01B0  ED 52                 SBC   HL,DE       ;Subtract the end from the found loc.
01B2  7C                    LD    A,H         ;
01B3  FE 00                 CP    00H         ;See if H is zero
01B5  20 05                 JR    NZ,NOPE     ;Go to NOPE if it isn't
01B7  7D                    LD    A,L         ;
01B8  D6 04                 SUB   04H         ;Correction factor
01BA  28 14                 JR    Z,NOMOVE    ;If zero, then don't move anything
01BC  2B          NOPE:     DEC   HL          ;Decrease HL by four
01BD  2B                    DEC   HL          ;
01BE  2B                    DEC   HL          ;
01BF  2B                    DEC   HL          ;
01C0  C5                    PUSH  BC          ;Keep BC for a while
01C1  E5                    PUSH  HL          ;Push HL
01C2  C1                    POP   BC          ;Byte count into BC
01C3  DD E5                 PUSH  IX          ;Place where vector was found
01C5  E1                    POP   HL          ;Put it in HL
01C6  23                    INC   HL          ;Add four to HL
01C7  23                    INC   HL          ;
01C8  23                    INC   HL          ;
01C9  23                    INC   HL          ;
01CA  DD E5                 PUSH  IX          ;Put the found location
01CC  D1                    POP   DE          ;into the destination register
01CD  ED B0                 LDIR              ;Move the vectors smoothly in place
01CF  C1                    POP   BC          ;Restore BC
01D0  FD 2B       NOMOVE:   DEC   IY          ;And close up the end
01D2  FD 2B                 DEC   IY          ;
01D4  FD 2B                 DEC   IY          ;
01D6  FD 2B                 DEC   IY          ;
01D8  79                    LD    A,C         ;Load MPAGE into A
01D9  21 1000               LD    HL,NUMVO    ;Load HL with the base address
01DC  85                    ADD   A,L         ;Add the offset
01DD  6F                    LD    L,A         ;
01DE  7E                    LD    A,(HL)      ;Get the number of vectors into A
01DF  3D                    DEC   A           ;Decrease the # of vectors by one
01E0  77                    LD    (HL),A      ;Save it back where it was found
01E1  5F                    LD    E,A         ;
01E2  78                    LD    A,B         ;Get the DPAGE
01E3  B9                    CP    C           ;Is it MPAGE?
01E4  20 04                 JR    NZ,ADONE    ;If not, jump to ADONE
01E6  7B                    LD    A,E         ;Otherwise restore A
01E7  32 1004               LD    (NUMVEC),A;And place the new number in NUMVEC
01EA  DD E1       ADONE:    POP   IX          ;Restore IX to it's orignal value
01EC  C3 00A1               JP    INPUT       ;Go back
```

Delete Command

```
                            SUBTTL Delete Command
01EF  79          DEL:      LD    A,C         ;Get the MPAGE
01F0  21 1000               LD    HL,NUMVO    ;Get the base address
01F3  85                    ADD   A,L         ;
01F4  6F                    LD    L,A         ;Put it back
01F5  7E                    LD    A,(HL)      ;Get the # of vectors
01F6  F6 00                 OR    00H         ;Is it zero?
01F8  CA 00A1               JP    Z,INPUT     ;If so, go back
01FB  3D                    DEC   A           ;Otherwise decrease the #
01FC  77                    LD    (HL),A      ;And put it back
01FD  57                    LD    D,A         ;
01FE  79                    LD    A,C         ;Load MPAGE in A
01FF  B8                    CP    B           ;Compare to DPAGE
0200  C2 0207               JP    NZ,DECR     ;Go back if not same
0203  7A                    LD    A,D         ;Get the # back
0204  32 1004               LD    (NUMVEC),A;And put in NUMVEC
0207  FD 2B       DECR:     DEC   IY          ;Decrease IY by four
0209  FD 2B                 DEC   IY          ;
020B  FD 2B                 DEC   IY          ;
020D  FD 2B                 DEC   IY          ;
020F  C3 00A1               JP    INPUT       ;Go back
```

Listing 1 continued on page 524

Listing 1 continued:

Reset Command

```
                              SUBTTL Reset Command
                   REST:      INWAIT            ;Get the next word
0212  37                      SCF               ;Set carry flag
0213  FB                      EI                ;Enable interrupts
0214  38 FE        ..000D:    JR    C,..000D    ;Wait on interrupt
0216  FE 0F                   CP    0FH         ;Is it the key?
0218  CA 0000                 JP    Z,0000H     ;If so, RESET
021B  C3 00A1                 JP    INPUT       ;If not, go back
```

Test Command

```
                              SUBTTL Test Command
021E  ED 5B 1005   TESTR:     LD    DE,(MEM)    ;Get the base address
0222  3A 1004                 LD    A,(NUMVEC);Save NUMVEC
0225  F5                      PUSH  AF          ;
0226  3E 26                   LD    A,26H       ;Load the # of vectors
0228  32 1004                 LD    (NUMVEC),A;
022B  21 0244                 LD    HL,MTEST    ;And the address of MTEST
022E  22 1005                 LD    (MEM),HL    ;
0231  F1                      POP   AF          ;Save A in H
0232  67                      LD    H,A         ;
                              INWAIT            ;Wait on next word
0233  37                      SCF               ;Set carry flag
0234  FB                      EI                ;Enable interrupts
0235  38 FE        ..000E:    JR    C,..000E    ;Wait on interrupt
0237  F5                      PUSH  AF          ;
0238  7C                      LD    A,H         ;
0239  ED 53 1005              LD    (MEM),DE    ;Restore display
023D  32 1004                 LD    (NUMVEC),A;
0240  F1                      POP   AF          ;
0241  C3 00A5                 JP    SKIP        ;And go back
```

Display Test Pattern

```
                              SUBTTL Display Test Pattern
0244  00 00 FF 00  MTEST:     DB    00H,00H,0FFH,00H        ;Large square
0248  FF 00 FF FF             DB    0FFH,00H,0FFH,0FFH
024C  FF FF 00 FF             DB    0FFH,0FFH,00H,0FFH
0250  00 FF 00 00             DB    00H,0FFH,00H,00H
0254  00 00 FF FF             DB    00H,00H,0FFH,0FFH       ;Diagonals
0258  FF 00 00 FF             DB    0FFH,00H,00H,0FFH
025C  80 FF 80 00             DB    080H,0FFH,080H,00H      ;Crosshairs
0260  FF 80 00 80             DB    0FFH,080H,00H,080H
0264  40 40 40 C0             DB    040H,040H,040H,0C0H     ;Small box
0268  40 C0 C0 C0             DB    040H,0C0H,0C0H,0C0H
026C  C0 C0 C0 40             DB    0C0H,0C0H,0C0H,040H
0270  C0 40 40 40             DB    0C0H,040H,040H,040H
0274  20 08 20 10             DB    20H,08H,20H,10H         ;Large T
0278  1C 10 24 10             DB    1CH,10H,24H,10H
027C  28 08 28 10             DB    28H,08H,28H,10H         ;Large E
0280  28 08 2C 08             DB    28H,08H,2CH,08H
0284  28 10 2C 10             DB    28H,10H,2CH,10H
0288  28 0C 2A 0C             DB    28H,0CH,2AH,0CH
028C  30 08 34 08             DB    30H,08H,34H,08H         ;Large S
0290  34 08 34 0C             DB    34H,08H,34H,0CH
0294  34 0C 30 0C             DB    34H,0CH,30H,0CH
0298  30 0C 30 10             DB    30H,0CH,30H,10H
029C  30 10 34 10             DB    30H,10H,34H,10H
02A0  3C 08 3C 10             DB    3CH,08H,3CH,10H         ;Large T
02A4  38 10 40 10             DB    38H,10H,40H,10H
02A8  60 08 60 0C             DB    60H,08H,60H,0CH         ;Small T
02AC  5E 0C 62 0C             DB    5EH,0CH,62H,0CH
02B0  64 08 64 0C             DB    64H,08H,64H,0CH         ;Small E
02B4  64 08 66 08             DB    64H,08H,66H,08H
02B8  64 0C 66 0C             DB    64H,0CH,66H,0CH
02BC  64 0A 65 0A             DB    64H,0AH,65H,0AH
02C0  68 08 6A 08             DB    68H,08H,6AH,08H         ;Small S
02C4  6A 08 6A 0A             DB    6AH,08H,6AH,0AH
02C8  6A 0A 68 0A             DB    6AH,0AH,68H,0AH
02CC  68 0A 68 0C             DB    68H,0AH,68H,0CH
02D0  68 0C 6A 0C             DB    68H,0CH,6AH,0CH
02D4  6E 08 6E 0C             DB    6EH,08H,6EH,0CH         ;Small T
02D8  6C 0C 70 0C             DB    6CH,0CH,70H,0CH
```

Get Character Interrupt

```
                              SUBTTL Get Character Interrupt
02DC  F3           GETCHR:    DI                ;Disable interrupts
02DD  DB 00                   IN    A,(00H)     ;Input byte
02DF  3F                      CCF               ;Complement the carry flag
02E0  ED 4D                   RETI              ;And return
```

Table Page

```
                              SUBTTL Table Page
                              ORG   0700H       ;Make room for a ROM table
0700  1007         TABLE:     DW    PAGE0       ;First free space in PAGE0
0702  1403                    DW    PAGE1       ;First free space in PAGE1
0704  17FF                    DW    PAGE2       ;First free space in PAGE2
0706  1BFB                    DW    PAGE3       ;First free space in PAGE3
                   ;
                                                ;                 Instruction #
0009               NUMCOM     EQU   9           ;Number of commands + 1
0708  00A1         JTABLE:    DW    INPUT       ;Pointer to INPUT        0
070A  00B8                    DW    MODP        ;Pointer to MODP         1
070C  00DD                    DW    DISP        ;Pointer to DISP         2
070E  010B                    DW    CLEAR       ;Pointer to CLEAR        3
0710  0122                    DW    DRAWV       ;Pointer to DRAWV        4
0712  015C                    DW    UNDRAW      ;Pointer to UNDRAW       5
0714  01EF                    DW    DEL         ;Pointer to DEL          6
0716  0212                    DW    REST        ;Pointer to RESET        7
0718  021E                    DW    TESTR       ;Pointer to TEST         8
                   ;
                              ORG   07FEH       ;Place for INRPTV
07FE  02DC         INRPTV:    DW    GETCHR      ;Save address of get character
                   ;
                              SUBTTL RAM Memory Allocation
                              ORG   1000H       ;Location of data
1000               NUMV0:     DS    1           ;Number of vectors in PAGE0
1001               NUMV1:     DS    1           ;Number of vectors in PAGE1
1002               NUMV2:     DS    1           ;Number of vectors in PAGE2
1003               NUMV3:     DS    1           ;Number of vectors in PAGE3
1004               NUMVEC:    DS    1           ;Number of vectors in current page
1005               MEM:       DS    2           ;Pointer to current page
1007               PAGE0:     DS    255*4       ;Room for 255 vectors--PAGE0
1403               PAGE1:     DS    255*4       ;Room for 255 vectors--PAGE1
17FF               PAGE2:     DS    255*4       ;Room for 255 vectors--PAGE2
1BFB               PAGE3:     DS    255*4       ;Room for 255 vectors--PAGE3
                              SUBTTL Macros And Symbols
                   ALL:       END   INIT        ;TTTHAT'S ALL FOLKS !!!!!!!!!
```

Macros And Symbols

```
Macros:
INWAIT

Symbols:
..0000  0090    ..0001  00A3    ..0002  00BA    ..0003  00DF
..0004  012F    ..0005  0133    ..0006  0137    ..0007  013B
..0008  0145    ..0009  0166    ..000A  016B    ..000B  0170
..000C  0175    ..000D  0214    ..000E  0235    ADONE   01EA
ALL     1FF7    CLEAR   010B    DECR    0207    DEL     01EF
DISP    00DD    DRAWV   0122    FIND1   0195    FIRST   008E
```

Listing 1 continued:

```
GETCHR  02DC   HERE    0140   INIT    0000   INPUT   00A1
INRPT   0066   INRPTV  07FE   JTABLE  0708   LOOP    0143
MEM     1005   MODP    00B8   MTEST   0244   NOFIND  0180
NOMORE  0083   NOMOVE  01D0   NOPE    01BC   NUMCOM  0009
NUMVO   1000   NUMV1   1001   NUMV2   1002   NUMV3   1003
NUMVEC  1004   OK      011B   PAGE0   1007   PAGE1   1403
PAGE2   17FF   PAGE3   1BFB   REST    0212   SKIP    00A5
TABLE   0700   TESTR   021E   UNDRAW  015C   WELL    0187

No  Fatal error(s)
```

Listing 2: *The BASIC program LCIRCLE, for the TRS-80, used to create photo 2.*

```
10 REM LCIRCLE BY BILLY GARRETT
20 DEFINT P,I,K,J,X,Y
30 DIM X(25),Y(25)
40 B=710/113
50 C1=255/2:C=C1
60 INPUT"INPUT NUMBER OF POINTS ON CIRCLE (3-23)";J
70 M=B/J:J1=J-1
80 IFJ>23ORJ<3 GOTO 60
90 INPUT"DRAW OR UNDRAW (D OR U)";S$
100 IF S$="D" THEN COM=4:GOTO130
110 IF S$="U" THEN COM=5:GOTO160
120 GOTO 90
130 INPUT"DO YOU WANT TO CLEAR THE PAGE FIRST (Y OR N)";A$
140 IFA$="Y" THEN OUTO,3:GOTO160
150 IF A$<>"N" GOTO 130
160 INPUT"WOULD YOU LIKE TO CHANGE THE RADIUS (10-127.5)";C
170 IFC>C1ORC<10THEN160
180 FORP=0TOJ:D=M*P:X(P)=C*(COS(D)+1)+C1-C:Y(P)=C*(SIN(D)+1)+C1-C:NEXTP
190 FORI=1TOJ1:FORK=I+1TOJ
200 OUTO,COM:GOSUB250:OUTO,X(I):OUTO,Y(I):OUTO,X(K):OUTO,Y(K):NEXTK,I
210 INPUT"ALL DONE. DO YOU WANT TO DO IT AGAIN (Y OR N)";A$
220 IFA$="N"END
230 IFA$="Y" GOTO 60
240 GOTO 210
250 IFINP(0)<128GOTO250ELSERETURN
```

Listing 3: *BASIC programs used to create images in photo 11.*

(3a)

```
10 REM GRID BY BILLY GARRETT
20 INPUT"ENTER STEP RATE (1-254)";S
30 IF S<2 OR S>254 THEN 20
40 INPUT"DRAW OR UNDRAW";A$
50 IF A$="D" THEN C=4 : GOTO70
60 IF A$="U" THEN C=5 : GOTO90
70 INPUT"CLEAR PAGE (Y) ";A$
80 IF A$="Y" THEN OUTO,3
90 FOR A=0 TO 255 STEP S
```

Listing 3 continued on page 526

Listing 3 continued:

```
100 GOSUB140:OUTO,C:GOSUB140:OUTO,A:OUTO,0:OUTO,A:OUTO,255
110 GOSUB140:OUTO,C:GOSUB140:OUTO,0:OUTO,A:OUTO,255:OUTO,A
120 NEXT A
130 GOTO 20
140 IFINP(0)<128THEN140ELSERETURN
```

(3b)

```
10 REM SPHERE BY BILLY GARRETT
20 DEFINT J,I,C,X,Y
30 DIM X(255),Y(255)
40 B=710/113
50 C=255/2
60 INPUT"INPUT NUMBER OF PONITS ON CIRCLE (3-254)";J
70 IFJ>254ORJ<3 GOTO 60
80 INPUT"SCAN OR DRAW ONCE (S OR D)";P$
90 IF P$="S" S=5 :GOTO120
100 IF P$="D" S=4:GOTO120
110 GOTO80
120 M=B/J
130 FORP=OTOJ:D=M*P:X(P)=C*(COS(D)+1):Y(P)=C*(SIN(D)+1):NEXTP
140 INPUT"DO YOU WISH TO CLEAR THE DISPLAY (Y)";A$:IFA$="Y"OUTO,3
150 IF S=5 THEN 180
160 FORI=OTOJ-1:OUTO,4:OUTO,128:OUTO,128:OUTO,X(I):OUTO,Y(I):NEXTI
170 GOTO 190
180 FORI=OTOJ-1:OUTO,6:OUTO,4:OUTO,128:OUTO,128:OUTO,X(I):OUTO,Y(I):NEXTI:GOTO180
190 INPUT"ALL DONE. DO YOU WANT TO DO IT AGAIN (Y OR N)";A$
200 IFA$="N"END
210 IFA$="Y" GOTO 60
220 GOTO 190
```

(3c)

```
10 REM COCENTER BY BILLY GARRETT
20 DEFINT J,I,X,Y
30 DIM X(255),Y(255)
40 B=710/113
50 C1=255/2
60 INPUT"CLEAR PAGE (Y)";Y$:IFY$="Y"OUTO,3
70 INPUT"INPUT NUMBER OF POINTS ON CIRCLES, AND NUMBER OF CIRCLES TO DRAW";J,N
80 IFJ*N>255GOTO 70
90 M=B/J
100 FOR C2=C1 TO 10 STEP -127.5/N
110 FORP=OTOJ:D=M*P:X(P)=C2*(COS(D)+1)+C1-C2:Y(P)=C2*(SIN(D)+1)+C1-C2:NEXTP
120 FORI=OTOJ-1:OUTO,4:OUTO,X(I):OUTO,Y(I):OUTO,X(I+1):OUTO,Y(I+1):NEXTI
130 NEXTC2
140 INPUT"ALL DONE. DO YOU WANT TO DO IT AGAIN (Y OR N)";A$
150 IFA$="N"END
160 IFA$="Y" GOTO 40
170 GOTO 140
```

(3d)

```
10 REM BOXES BY BILLY GARRETT
20 DEFINT G,C,A
30 FOR G=64 TO 2 STEP-1
40 FOR C=4 TO 5
50 FOR A=0 TO 127 STEP G
60 B=255-A
70 GOSUB130:OUTO,C:OUTO,A:OUTO,A:OUTO,B:OUTO,A
80 GOSUB130:OUTO,C:OUTO,B:OUTO,A:OUTO,B:OUTO,B
90 GOSUB130:OUTO,C:OUTO,B:OUTO,B:OUTO,A:OUTO,B
100 GOSUB130:OUTO,C:OUTO,A:OUTO,B:OUTO,A:OUTO,A
110 NEXTA,C,G
120 GOTO 30
130 IFINP(0)<128THEN130ELSERETURN
```

(3e)

```
10 REM HOURGLASS BY BILLY GARRETT
20 INPUT"ENTER STEP RATE (1-254)";S
30 DEFINT A
40 OUTO,3
50 FOR A=0 TO 254 STEP S
60 OUTO,4:OUTO,A:OUTO,0:OUTO,255-A:OUTO,255
70 NEXT
80 GOTO 20
```

(3f)

```
10 REM SOLIDSIN BY BILLY GARRETT
20 M=710/113
30 INPUT"STEP,CYCLE";Q,B
40 INPUT"DO YOU WISH TO CLEAR THE DISPLAY (Y)";A$:IFA$="Y"OUTO,3
50 FOR A=1 TO 255 STEP Q
60 OUTO,4:OUTO,A:OUTO,127.5*(SIN(A*M/B)+1):OUTO,A:OUTO,0:NEXT
70 INPUT"DO YOU WANT TO TRY AGAIN (Y)";A$:IF A$="Y" THEN 20 ELSE END
```

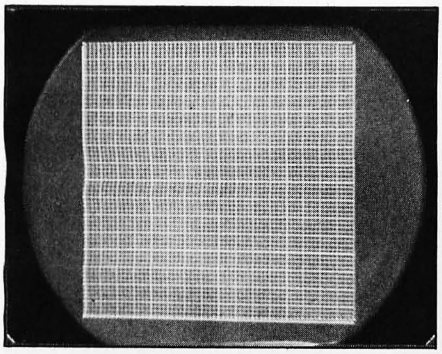
(3g)

```
10 REM SSINE BY BILLY GARRETT
20 DIM X(511)
30 M=355/113*2/255: C=127.5: B=1
40 FOR A=0 TO 511
50 X(A)=C*(SIN(A*M)+B)
60 NEXT A
70 INPUT"DO YOU WISH TO CLEAR THE DISPLAY (Y)";A$:IFA$="Y"OUTO,3
80 INPUT"STEP RATE (1-254)";S
90 FOR A=0 TO 255 STEP 255/S
100 IF A+S>255 GOTO 120
110 OUTO,4:OUTO,A:OUTO,X(A):OUTO,A+S:OUTO,X(A+S)
120 NEXT A
130 GOTO 70
```

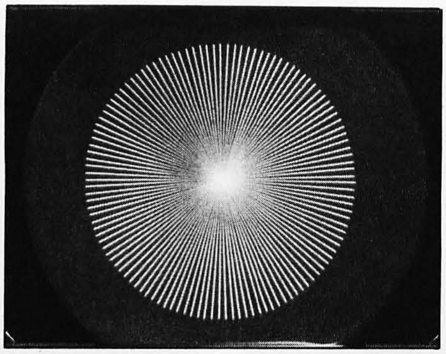
Listing 4: *The BASIC program STICKS used to create photo 3.*

```
10 REM STICKS BY BILLY GARRETT
20 DEFINT A,S,X
30 DIM X(514)
40 M=355/113*2/255
50 FOR A=0 TO 513
60 X(A)=127.5*(SIN(A*M)+1):NEXT A
70 PRINT"HERE WE GO!!!"
80 FOR S=6 TO 256:OUTO,1:OUTO,S:OUTO,2:OUTO,S-1:OUTO,3
90 FOR A=0 TO 255 STEP 255/S
100 IF A+S > 255 THEN Q=510-(A+S) ELSE Q=A+S
110 OUTO,4:OUTO,X(A):OUTO,X(512-A):OUTO,Q:OUTO,X(A+S)
120 NEXT A,S
```

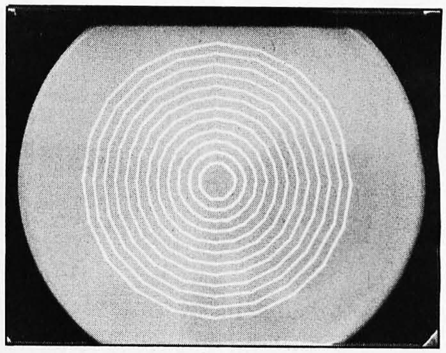
(11a)



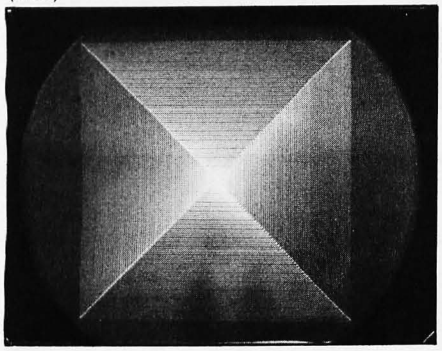
(11b)



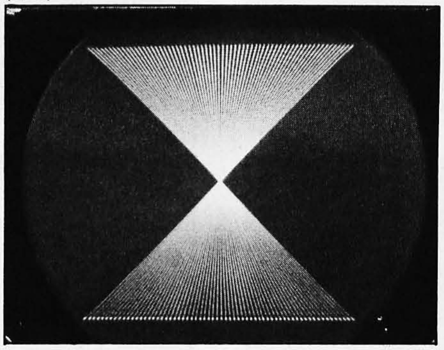
(11c)



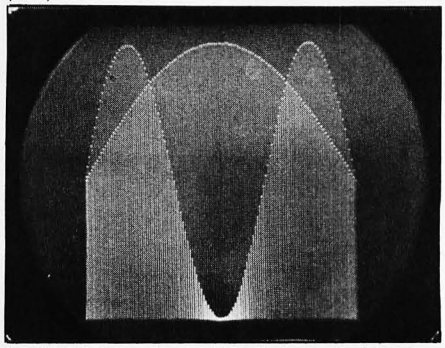
(11d)



(11e)



(11f)



(11g)

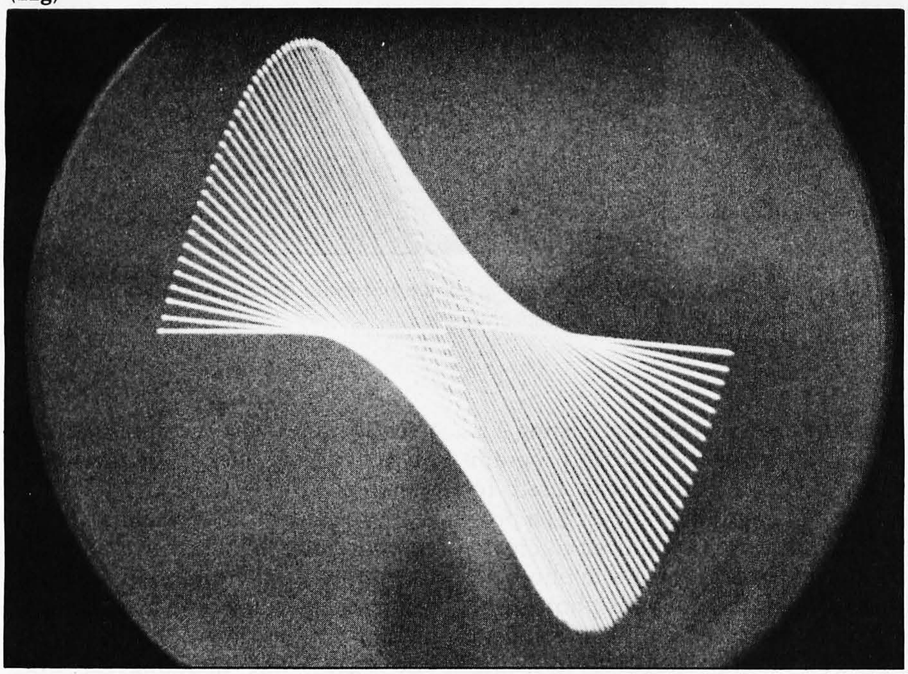


Photo 11: *Examples of the elementary images that can be produced by Microvec. Each of these images was produced by a corresponding program in listing 3: photo 11a is a drawing produced by GRID, a BASIC program for the TRS-80 (see listing 3a); photo 11b was drawn by a program called SPHERE (listing 3b); photo 11c is the product of COCENTER (listing 3c); photo 11d is an image created by BOXES (listing 3d); photos 11e, 11f, and 11g were produced by HOURGLASS (listing 3e), SOLIDSIN (listing 3f), and SSINE (listing 3g), respectively.*

Text continued from page 520:

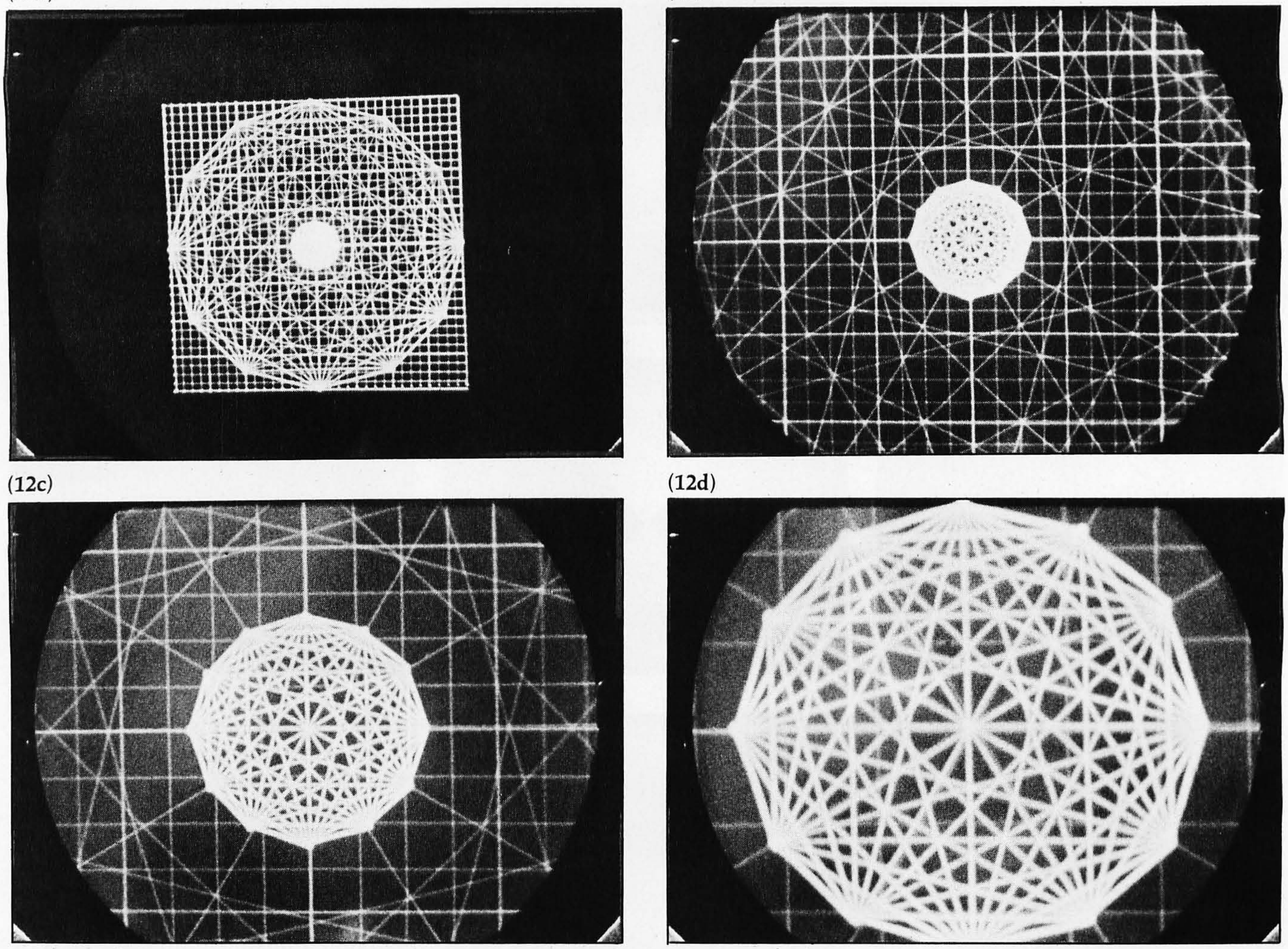
that, adjust R18 and R26 to about 8.5 kΩ. Finally, adjust R10 to about 9 kΩ. Now, plug in the ICs, plug your board in, and hook the outputs up to your scope. Set the x and y channels of the scope to about 0.5 V/cm. Finally, turn on the power. If everything is working properly, you should be able to see the board trying to display a few vectors on the screen. Eventually, these will look like the image in photo 8. If nothing happens, or you are having problems, go on to the next section, concerning Murphy's Law.

Once you obtain some kind of display, adjust the gain and zero controls of each channel until you get an output signal roughly ±1 V centered about ground (see photo 10). Then, adjust the vector controller's slope controls until you see the diagonals intersect the middle. Next, vary the endpoint adjustment until the ends of the boxes just touch. You will have to play with the slope and endpoint adjustments for a while until the display looks right. Once you get your display looking like photo 8, you are almost finished. Next, increase the gain of each channel until the display begins to come apart. Then, back the gain down just a bit and adjust your oscilloscope so that the display fills the screen and the brightness is to your liking. If you get this far, you are doing very well.

Next, verify that all the commands work as described. A good check of the circuit board would be to run some of the programs in the later section of examples.

What to Do if Murphy's Law Is Enforced

You probably know the old saying, "If anything can go wrong, it will." This project will certainly be no exception. Refer to photos 4, 5, 6, and 7 if you are having problems. Read over the circuit description and make sure that the proper signals are at the given pins. For example:



(12a) (12b) (12c) (12d)

Photo 12: *Using special features of the oscilloscope to alter images. Here, an image is shown normally (photo 12a) and with fourfold, sixteenfold, and hundredfold magnification (photos 12b, 12c, and 12d, respectively).*

- If you are able to get a test pattern but it will not respond to commands, check to see that the data word is being strobed into the PIO when your processor does an I/O operation.
- If you suspect that the Z80 is running the wrong code, see if the NMI request pin is being pulsed as shown in photo 4.
- If the memory sections are not hooked up properly, you will not be getting the four short pulses on pin 15 of IC18, as is shown in photo 5.
- If the NMI pin of the Z80 does not remain high during and shortly after reset, the processor will almost certainly begin executing at the wrong address. Since R7 pulls the NMI line high during reset, make sure it is installed.

A Few Examples

Although the commands are fairly simple to understand, I have developed some examples that are both instructive and entertaining. Listing 2 is a TRS-80 Level II BASIC program called LCIRCLE that was used to draw the images in photo 2. The programs in listing 3 were used to create the corresponding images in photo 11 (e.g., listing 3b produced photo 11b, listing 3c produced photo 11c, and so on). The program STICKS, in listing 4, is the one that drew photo 3. The two parts of photo 1 are freehand "doodles" done with a program called SKETCH (not listed in this article).

Photos 12a through 12d show an interesting feature of the vector-graphics display. By simply increasing the gain on the oscilloscope, you can magnify the frame. The first photo is at the normal setting; the second is magnified by 4; the third is magnified by 16; and the last is magnified by 100 times. Notice that the little circle that was inside the larger circle is blown up so that it is larger than the original circle. All this magnification was accomplished simply by varying the gain on the oscilloscope.

If you want to try animating a complicated scene, rather than trying to redraw each frame: first, put the same background in two different memory pages; then, by displaying one page while updating the next, you can swap the displayed pages back and forth to achieve the appearance of rapid motion. Your best teacher, though, will be experience.

Conclusion

This board is an exciting peripheral device for any computer. The first time you see these displays produced by your computer, you will not believe how sharp and detailed they are. Many creative uses for this display will be possible. I welcome suggestions or comments, but please enclose a SASE if you want a reply. Happy drawing! ■