Build the Micro D-Cam Solid-State Video Camera

Part 2: Computer Interfaces and Control Software

Serial interfaces for the Apple II and the IBM Personal Computer and versatile software for the Apple II

by Steve Ciarcia

Last month I introduced you to the Micro D-Cam, a relatively low-cost direct-output digital camera that you can build, either from scratch or from a kit distributed by The Micromint. Using a 64K-bit dynamic memory chip as its optical sensor, it has a resolution of 256 by 128 pixels (picture elements), which is adequate for many applications, including input of graphic images, pattern and character recognition, robotics, process control, and security.

In part 1 I explained the principles of operation of the IS32 Optic RAM (random-access read/write memory) and the rest of the Micro D-Cam's hardware. (Table 1 may help you recall some of the IS32's characteristics.) This month I'd like to finish the project by discussing how the camera can be attached to the expansion buses of the Apple II Plus and the IBM Personal Computer and how the camera is programmed to work.

The amount of software included with this article is somewhat more than you've come to expect from a hardware-type fellow like me, but I feel it is necessary to properly show how software can be used to enhance the final picture. In particular, some of you may be interested in the method used to present a gray scale on an Apple II computer.

A Quick Review

The IS32 Optic RAM from Micron Technology Inc. is a memory chip specially packaged to function as a digital image-sensing device. (Because its output is a pure digital signal, it cannot be used to directly drive a composite-video monitor.) The IS32 contains 32,768 usable light-sensitive elements arranged in a matrix of 128 rows and 256 columns. Each of the elements in the matrix is a light-sensitive capacitor, a memory cell that can be accessed randomly by simply reading in the appropriate row and column address. Light striking a particular element causes the capacitor, which is initially precharged to a fixed voltage, to discharge toward 0 volts (V). The capacitor discharges at a rate proportional to the light intensity throughout the duration of the exposure. When the cell's content is read, a logic 0 remaining in the cell indicates a bright pixel—the capacitor was exposed to a light intensity sufficient to discharge the capacitor past the threshold point. A dark pixel is indicated by a logic 1 remaining in the cell, which happens when the light intensity is not sufficient to discharge the capacitor past the threshold point.

1. two 128- by 256-element arrays each measuring 5.504 by 1.088 millimeters
2. element size: 8 microns by 9 microns
3. vertical center-to-center spacing: 21.5 microns
4. horizontal spacing: 8.5 microns
5. spacing between left and right arrays: 150 microns

Table 1: *Specifications of the Micron Technology IS32 Optic RAM, a 64K-bit memory chip that has the extra talent of serving as a digital image detector.*

The operation of the image sensor can be compared to the function of film in a camera. The user can regulate the exposure by two adjustments: aperture (f-stop) and shutter speed. The aperture adjustment controls the amount of light that is allowed to expose the light-sensitive medium (either the IS32 or the film emulsion) by mechanically widening or narrowing the hole through which the light passes. The shutter speed (or scanning speed in the case of the IS32) dictates the amount of time the sensitive medium is exposed.

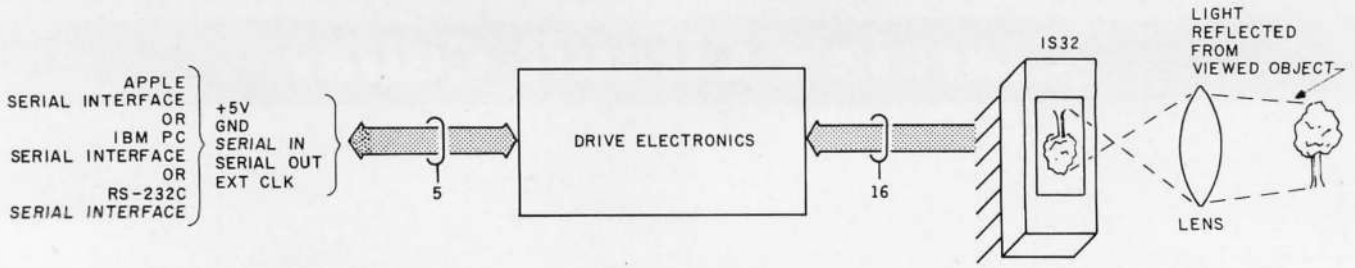


Figure 1: *A block diagram of the Micro D-Cam system.*

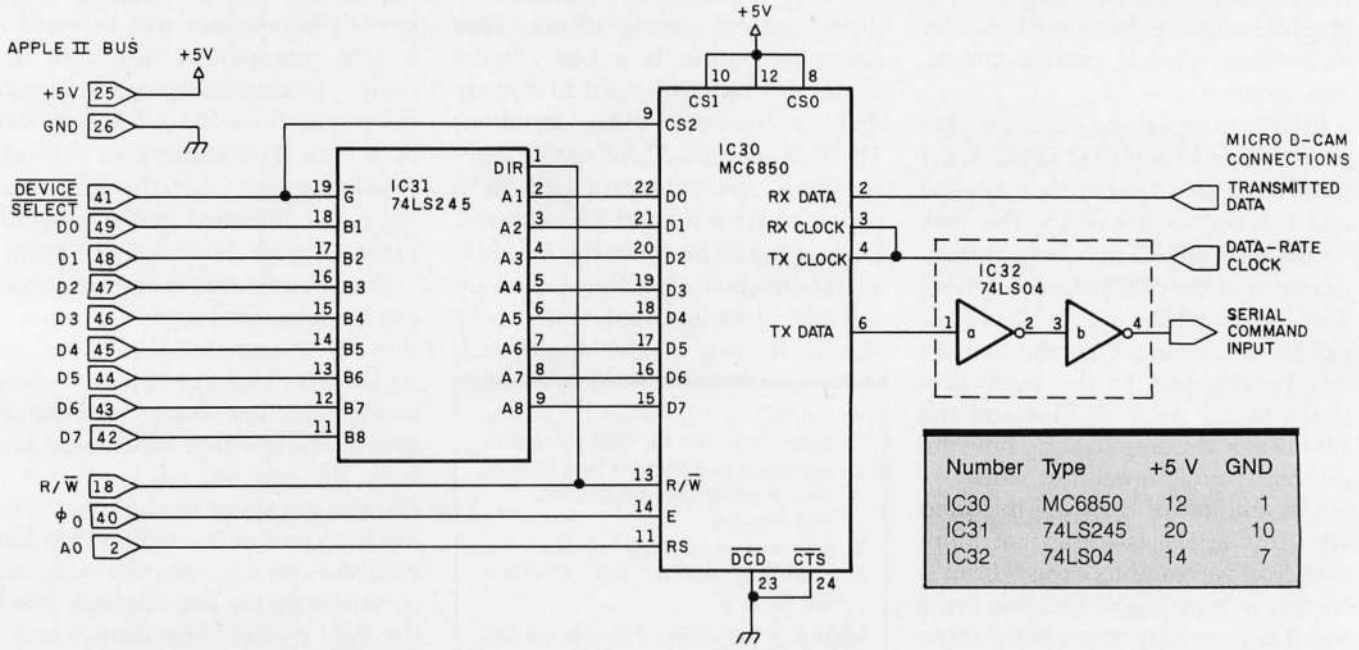
The Micro D-Cam's equivalent of an electronic shutter is controlled by commands transmitted to the interface. Sending a SOAK command to the Micro D-Cam has the effect of opening the shutter. After the appropriate period of exposure has elapsed, two commands, REFRESH and SEND, stop the exposure (close the shutter) and transmit the image to the host computer.

Interfacing the Micro D-Cam

Last month, when we looked at the control and driver electronics of the basic Micro D-Cam, we found that it communicates with its host computer serially, one bit at a time. In its minimal configuration, it requires four wires to be connected to the host computer: two supplying +5 V and ground potential and one each for serial data in and out. In a nonspecific configuration, it can operate asynchronously over an RS-232C link (at a data rate of up to 19,200 bps or bits per second), but I have devised serial interfaces for the camera that can be attached directly to the IBM PC and Apple II computers' buses (although still communicating serially). Using a fifth signal, an additional external clock signal provided to the bus interfaces by the drive electronics, the Micro D-Cam can then function at data rates up to 153,600 bps. The complexity of interface circuits of this type depends upon the host computer's bus structure and address range. The general scheme of connection is shown in figure 1.

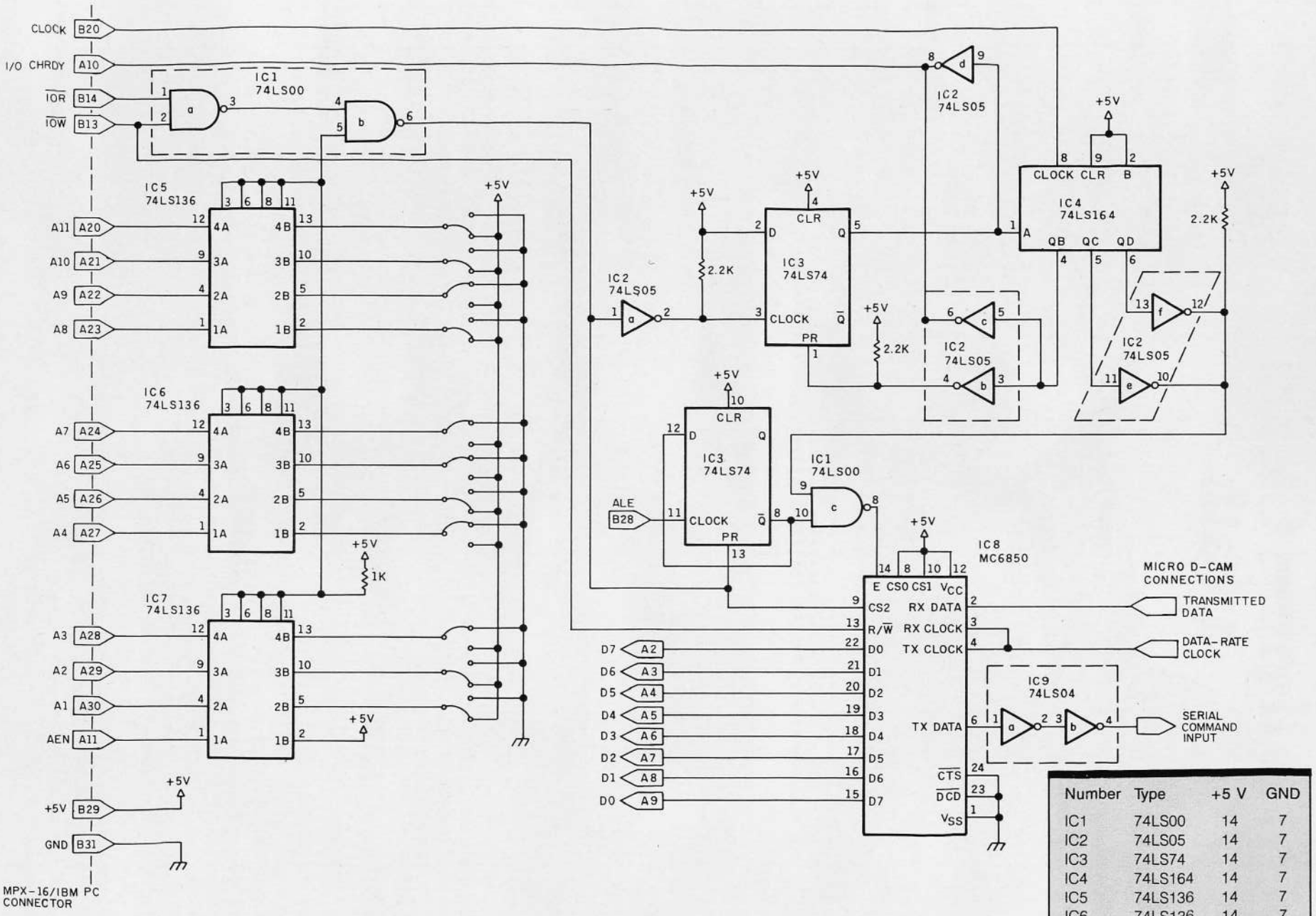
Figure 2 is a schematic diagram of the circuit that forms the interface from the Micro D-Cam circuitry (shown in part 1) to the expansion bus of the Apple II Plus computer. It owes its simplicity to the predecoding of the I/O (input/output) slot address already provided on the Apple's main circuit board. The address decoders usually required in a peripheral interface are eliminated, and the complete serial interface can be built with only two integrated circuits. The 74LS245 octal bus transceiver buffers the TTL- (transistor/transistor logic) level serial data into and out of the MC6850 ACIA (asynchronous communication interface adapter). The serial bit rate is controlled by the external clock output from the Micro D-Cam drive electronics. For maximum speed, the clock frequency should be set for 153,600 Hz.

Figure 3 on page 70 shows the serial interface circuit for the Micro D-Cam configured for the IBM PC's bus. Due to the greater complexity of the Intel 8088 processor as compared with the Apple's 6502 and the PC's larger memory-address space, the in-



Number	Type	+5 V	GND
IC30	MC6850	12	1
IC31	74LS245	20	10
IC32	74LS04	14	7

Figure 2: *A schematic diagram of an Apple II Plus or Apple IIe interface for the Micro D-Cam. The serial data stream from the Optic RAM is converted to parallel bytes and placed on the Apple's data bus by the ACIA and bus transceiver. Although operating asynchronously, high data rates (up to 153,600 bps) are possible because of the external data-rate clock input from the camera-control circuitry.*



Number	Type	+5 V	GND
IC1	74LS00	14	7
IC2	74LS05	14	7
IC3	74LS74	14	7
IC4	74LS164	14	7
IC5	74LS136	14	7
IC6	74LS136	14	7
IC7	74LS136	14	7
IC8	MC6850	12	1
IC9	74LS04	14	7

Figure 3: *An interface from the Micro D-Cam to an IBM Personal Computer (or a bus-compatible unit like the MPX-16). A set of jumper connections on the inputs of exclusive-OR gates determines the I/O-bus address of the interface, shown here set to hexadecimal xD26 and xD27.*

(2a)	Status Bit	Meaning When Set to 1
	0	data has been received from the camera
	1	a command may be sent to the camera
	2	unused
	3	unused
	4	received data was improperly framed
	5	data received before previous byte read

(2b)	Command Bit	Meaning When Cleared to 0
	7	none (always 1)
	6	none (always 1)
	5	alternating-bit mode (ALTBIT)
	4	wide-pixel mode (WIDEPIX)
	3	7-bit data bytes (7BIT)
	2	transmit one frame instead of two (1ARRAY)
	1	refresh instead of soak (REFRESH)
	0	send the requested image (SEND)

Table 2: *Meanings of bits in the status register (2a) and command word (2b) for the Apple II/Micro D-Cam interface.*

terface requires three times as many integrated circuits. In the IBM, the Micro D-Cam's two port addresses are decoded by three chips: IC5, IC6, and IC7. These are 74LS136 open-collector exclusive-OR gates connected together in a "wired-OR" configuration. The voltages wired to the 11 inputs of the address decoder determine the interface board's addresses. As shown in figure 3, the addresses I used were *x*D26 and *x*D27 (where *x* can take on any hexadecimal value from 0 to F). The 6850 ACIA (IC8) functions as previously described except that IC2 and IC4 are configured as a wait-state generator to facilitate timely access to the bus.

Data and Command Format

The 6850 ACIA comprises a data register and a status register. You can configure operating parameters (such as parity, stop bits, start bits, clocking, etc.) by writing values into the status register. Before the host computer can access the Micro D-Cam, the ACIA has to be initialized to the proper configuration. The control software does this by writing two bytes, a hexadecimal 03 followed by a hexadecimal 14, into the status register. The first byte performs a master reset on the ACIA, while the second byte specifies that the serial transmission protocol is 1 start bit, followed by 8 data bits, followed by 1 stop bit.

Reading the status register allows the control program to determine when new data has been received and when the ACIA is ready to send data. The meanings of the status bits, when set, are as shown in table 2a.

In normal use, only bit 0 is checked when seeing if data is available from the camera. Bits 4 and 5 are used only in debugging, as these situations should not normally arise. When designing the program that receives the image from the camera, it is a good idea to incorporate a time-out mechanism in case the camera stops sending bytes before the program expects; otherwise, the program can hang up if the software misses even a single byte.

In the Apple II Plus and IIe, the hexadecimal addresses of the type C0*n*E access the status register of the ACIA on an interface card plugged into the corresponding slot, while C0*n*F addresses access the ACIA's data register. The *n* is the hexadecimal value of the slot number plus 8. For example, suppose the interface card were plugged into slot 3; 3 plus 8 equals B, and so address C0BE will access the status register and C0BF the data register.

Command Functions

While the camera is running, the host computer directs the Micro D-Cam's operating modes by sending it command words. Each command word is composed of 8 bits, with functions as summarized in table 2b. Let's look at each of these in detail:

ALTBIT Mode: When bit 5 is clear (equal to 0), the Micro D-Cam transmits only the pixels from the even-numbered rows and columns in the Optic RAM. This mode usually produces a clearer image than the NOALTBIT mode at the expense of losing resolution.

WIDEPIX Mode: When bit 4 is clear, the Micro D-Cam transmits each pixel in the array twice. Each image-sensing element is rectangular in shape, so by "double-transmitting" the pixels, the proper width-to-height (aspect) ratio is maintained when the image is displayed on the computer's video monitor.

7BIT Mode: The Apple II's implementation of high-resolution graphics is somewhat peculiar. The most significant bit of each byte on the hires graphics page is reserved as the color bit for a group of pixels, while each of the other 7 bits stores a 1 or 0 as a bright or dark value for a pixel. In 7BIT mode, the Micro D-Cam transmits data in a format compatible with the Apple's high-resolution format, with 7 bits of pixel values per byte. The 7BIT mode is selected by clearing bit 3 of the command byte to 0. The alternative to 7BIT mode is 8BIT mode, which is achieved by setting bit 3 to 1. The 8BIT mode causes the camera to transmit in normal bit-mapped format, with all 8 bits in the byte containing image data, and is preferred for use with all computers other than the Apple.

1ARRAY Mode: The 1ARRAY mode is selected by clearing bit 2 of the command byte. Using this mode, only data from the image focused on the lower light-sensitive array is transmitted from the Micro D-Cam. By setting bit 2 of the command byte, 2ARRAY mode is selected, which causes data from both arrays to be transmitted from the camera. The 2ARRAY mode causes a split-screen effect because of the space between the two arrays in the image-sensor chip.

REFRESH Mode: In some ways, the Micro D-Cam is like any other

Command Character	Control Effect
>	increase exposure time
<	decrease exposure time
F	fix exposure time to current setting
L	load previously stored image from disk
N	print negative of screen image onto Epson printer
P	print screen output onto Epson (Graftrax option required)
Q	quit and return to main menu
R	toggle display of exposure time and light level
S	save current image to disk
T	use current light level and autotrack the exposure

Table 3: *Options for control of the Micro D-Cam that may be selected in real time through the distributed menu-driven software. See table 4 for the options provided in the GREY16 program.*

camera. It must receive the proper amount of light to make the image develop properly. Too much light will overexpose the image, while too little light will underexpose the image. Exposure time is determined by how long the control program in the host computer allows the Optic RAM to be exposed to light without its cells being refreshed. Refreshing the image sensor is the same process used in any dynamic memory: the existing charge in each cell is sensed, the voltage compared with a threshold potential, and a fresh potential of 0 V (for a logic 0) or +5 V (for 1) is rewritten into the cell. (The only difference in the Optic RAM is that all cells must contain +5 V at the beginning of an image-sensing cycle when refreshing stops.) If the image sensor is not continually refreshed, the light focused on each cell causes the voltage in each cell to leak away at a rate proportional to the intensity of the light. When the image sensor is not being refreshed, we say it is "soaking" (in light). Allowing the image sensor to soak for longer periods of time enables the Micro D-Cam to see better in dimmer light.

When the REFRESH mode is selected (by clearing bit 1 of the command byte) the Micro D-Cam keeps the image sensor's cells refreshed while it is sending an image. When bit 1 is set, SOAK mode is invoked. This causes the camera to soak (and therefore remain sensitive to light) while it is transmitting an image.

SEND Mode: When a command is sent to the camera with SEND mode selected (bit 0 cleared), the camera begins transmitting an image.

Control Software

The software for controlling and displaying pictures is vital to the operation of the Micro D-Cam. Menu-driven versions of the Micro D-Cam control software for both the Apple II and IBM PC are available from The Micromint.

However, some of you may already have the Micron Technology Optic RAM or a similar 64K-bit dynamic RAM device with suitable chip layout, and you may want to build the Micro D-Cam from scratch. Consequently, I have included with this article complete listings of two control programs written for the Apple II. One provides experimenters with a means for testing the Micro D-Cam; the second is a more sophisticated software routine that enhances the image and performs gray-scale ordered dithering (I'll explain this term later). While the Micro D-Cam software includes some additional menu-driven utility programs (some options of which are shown in table 3), all the Micro D-Cam photos printed here and last month can be reproduced using only the two programs in the magazine.

A Sample Control Program

The Micro D-Cam demonstration program (listing 1 on pages 512 through 518) illustrates the simplest possible software needed to receive an image from the camera and display it on the Apple's hi-res screen. It is not really as long and complicated as it looks; the accompanying flowchart (figure 4 on page 76) should reveal the general scheme of operation. The software consists of two parts: a short BASIC main program (listing 1a) and a set of machine-language subroutines (shown in assembled format in listing 1b). The BASIC program loads the machine-language code from disk, interactively sets the correct I/O-slot number and exposure time, and calls the machine-language code to display the image; upon returning to BASIC, the calling program checks to see if you want to terminate the process.

The hard part of the work is done in the machine-language routines, which were necessary to allow the Micro D-Cam to operate at 153,600 bps. When called, the machine code begins by making sure that the hi-res screen is being displayed. It then initializes the ACIA and sends a command to tell the camera to soak without sending an image. (This effectively clears the Optic RAM and tells the camera to begin the exposure.) The program then waits for the duration of the exposure.

The next step is to read the image from the camera and display it on the screen. To save time and memory, the software sends the picture straight to the hi-res screen memory (rather than reading it into a separate buffer area and then moving it) to minimize the processing of the final image. The mode used is alternate-pixel, wide-pixel, with 7-bit data words. Before any part of the picture is received, a number of memory pointers are set up to facilitate proper placement on the screen. A command is sent to the camera to begin transmitting the image, and the program loops to read in each byte of the image and put it on the screen.

The control software knows how many bytes of image data it should receive from the camera, but a problem can arise from relying on byte-counting to determine when to stop reading data: if the computer misses one, it could hang the system up. To be on the safe side, a time-out loop has been provided in the image-reading routine. If the computer times out

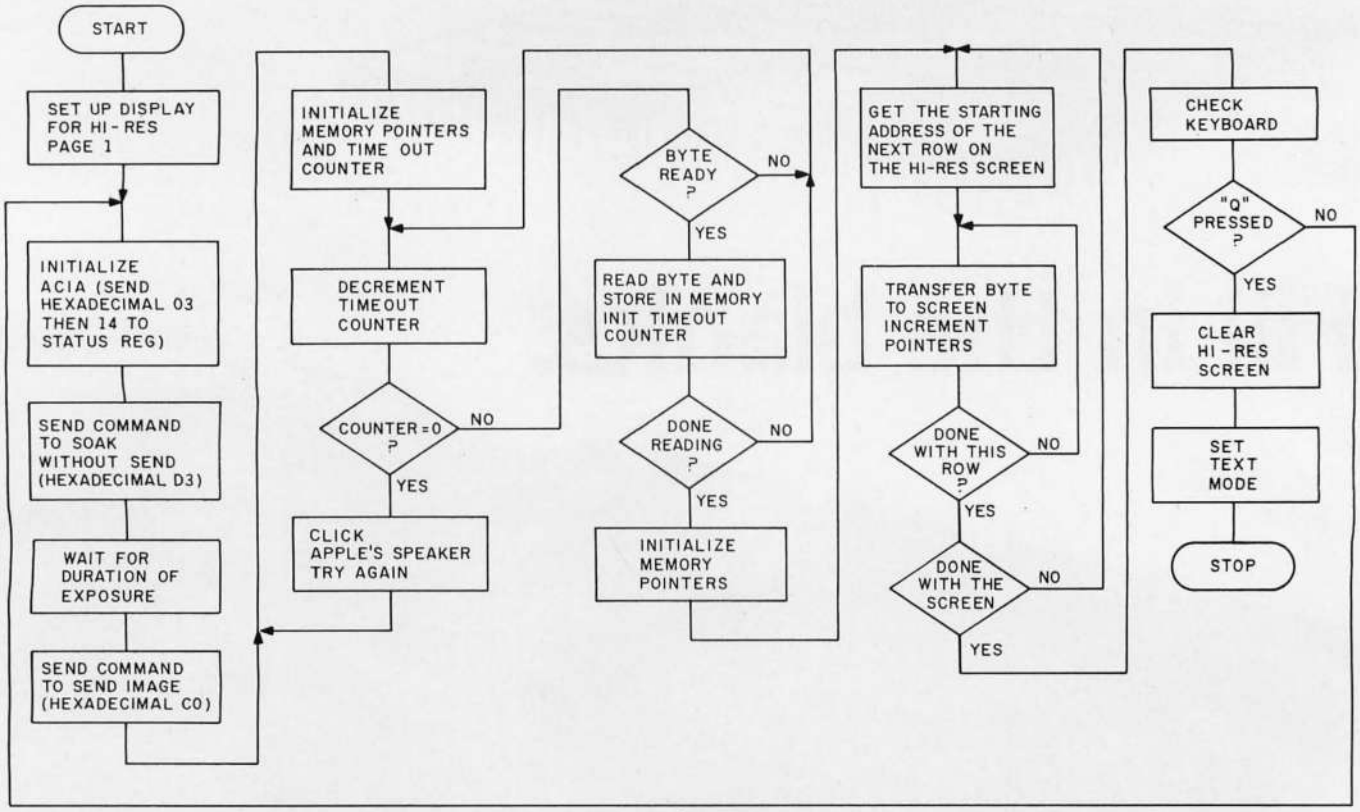


Figure 4: *A flowchart of the Micro D-Cam demonstration program for the Apple II. The program consists of a BASIC main routine, shown in listing 1a, and some 6502 machine-language subroutines, shown in assembly-language form in listing 1b.*

while waiting for the camera, it clicks the speaker, checks for a keypress, and tries the entire command sequence again. In this manner, you are alerted to any possible problems.

Because the Apple's hi-res screen display is mapped nonlinearly into memory space, a lengthy table at the end of the machine-language code provides the starting address for each consecutive row of the hi-res screen. The program gets the address of the beginning of each row and then reads 40 bytes from the camera, placing them consecutively on the screen. The next row, and each row after it, is done in a similar manner.

Once the image is on the screen, a command is sent to the camera to refresh without sending. This gets it ready for the next exposure. Finally, the machine code checks the keyboard and processes any command inputs before returning to BASIC.

Obtaining Gray Scale

A more user-friendly demonstration of the Micro D-Cam that also provides a level of gray-scale capability is the GREY16 program of listing 2 (pages 518 through 538). It has one mode that allows you to do quick aiming and focusing of the camera, another to let you get an idea of what the final picture will look like, and a third to create a 15-intensity-level gray-scale picture on the Apple II. (The processes involved are outlined in the flowchart of figure 5. Unfortunately, space constraints prevent me from showing you a similar program for the IBM PC.)

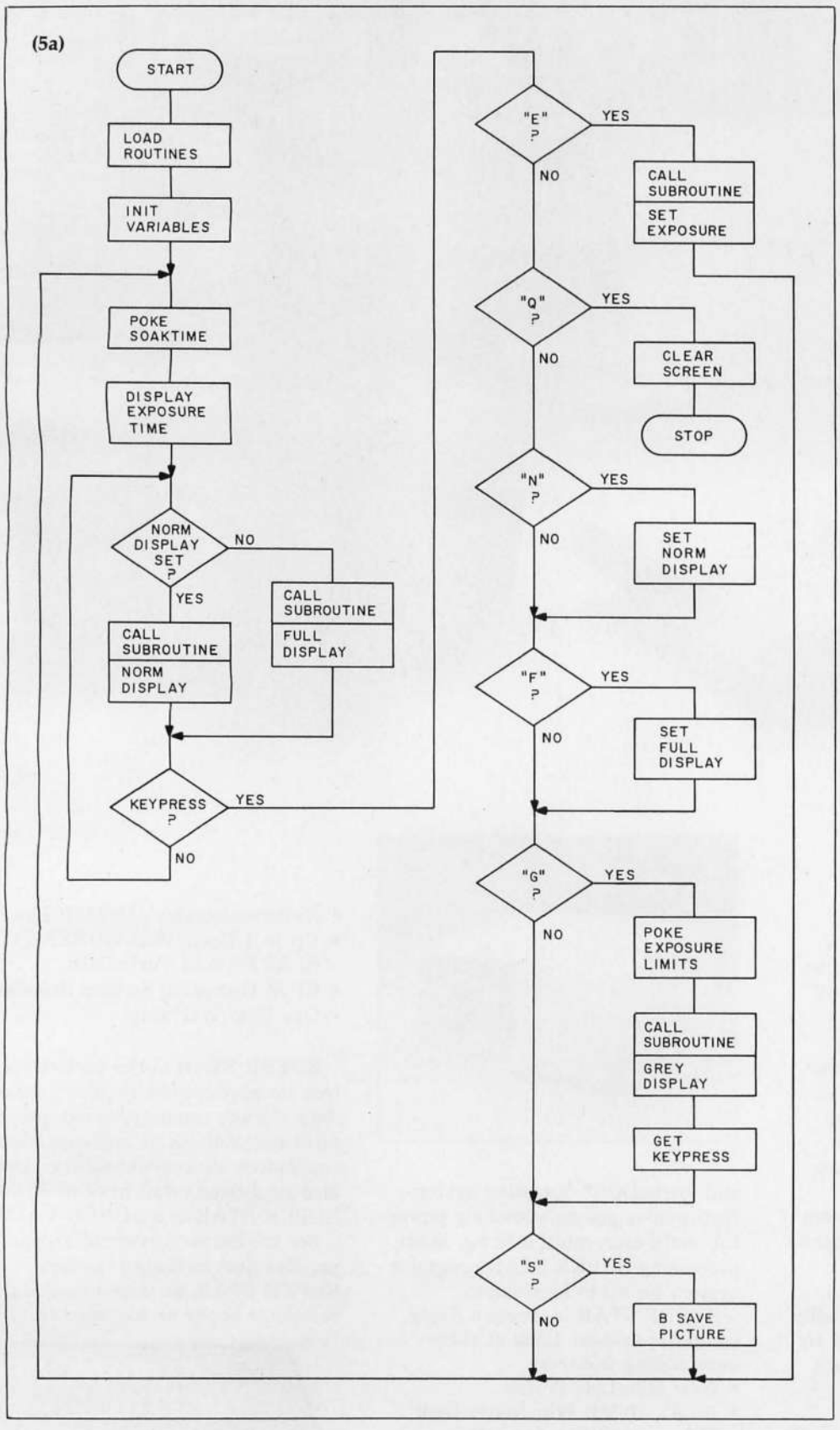
Using GREY16, you can change the length of exposure for the image being displayed, or you can change the upper and lower exposure limits of the gray-scale image. Once you've obtained a satisfactory picture, you can save it on disk for later use or print it on an Epson MX-80 printer (equipped with Graftrax) using the screen-dump program. A summary of available commands in the GREY16 program is shown in table 4.

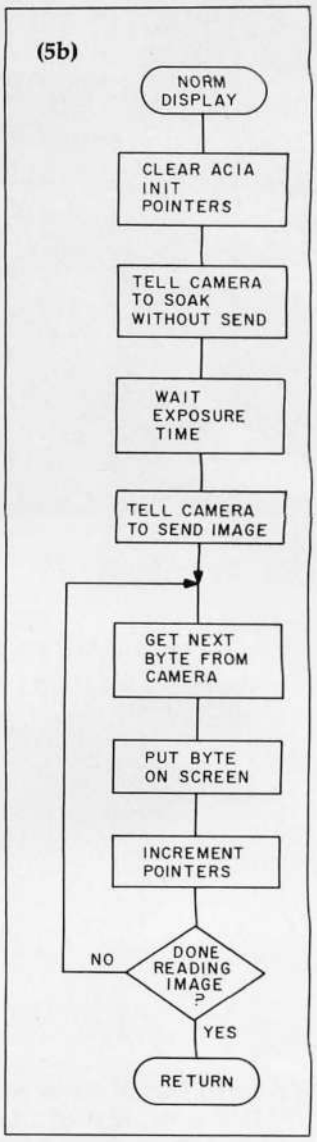
When it is first powered up, you start the camera running by selecting one of the options from the GREY16 menu. If the exposure time is insufficient, the screen will be black. If the exposure time is excessive, the screen

Command Character	Control Effect
N	display the image in normal size (256 by 64)
F	display the image in full size (256 by 128)
G	create a picture (256 by 128) with 15 levels of gray (this process takes about 30 seconds and displays a countdown of the number of exposures from F to 0)
E	change the exposure time of the current displayed image, the upper limit of the gray-scale image, or the lower limit of the gray-scale image
S	save to disk the picture currently being displayed (this may be done in any of the three display modes: normal, full, or gray)
Q	quit the program and return to BASIC

Table 4: *A summary of user commands implemented in the GREY16 program of listing 2.*

Text continued on page 82





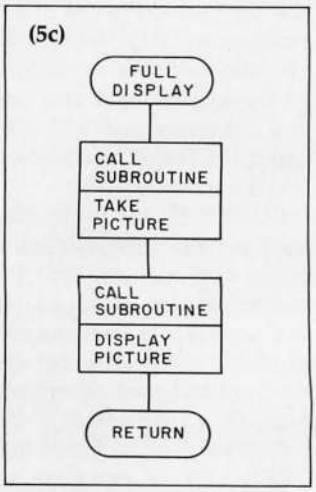
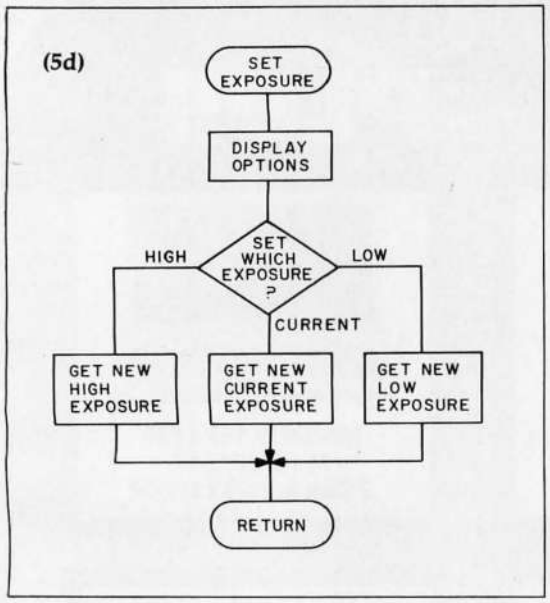


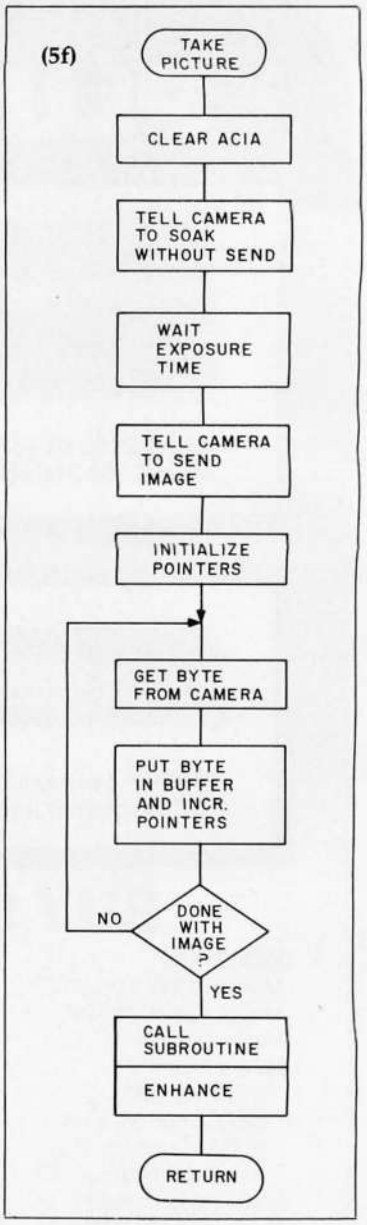
Figure 5: *Flowcharts of the GREY16 program for the Apple II (the figure continues on pages 80 and 82). The BASIC portion appears as listing 2a, the machine-language portion as listing 2b. The main routine (5a) calls various subroutines: NORM DISPLAY (5b), FULL DISPLAY (5c), SET EXPOSURE (5d), GREY DISPLAY (5e), TAKE PICTURE (5f), DISPLAY PIC (5g), and ENHANCE (5h).*

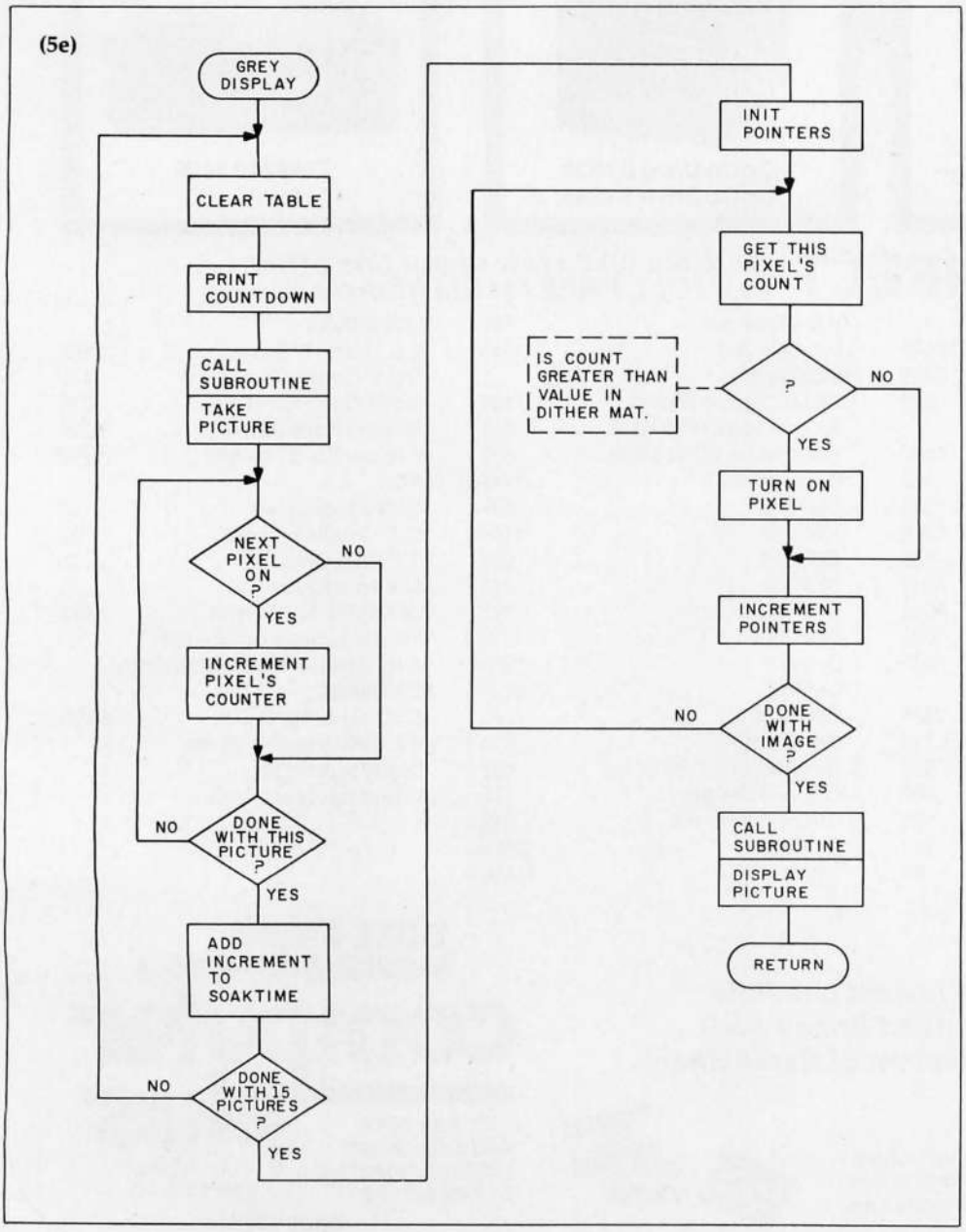
The subroutine GREY DISPLAY takes sensor pixels from 15 exposures and translates them into arrays of the smaller display pixels to represent intermediate brightnesses.

Figure continued on page 80

Figure 5 continued:







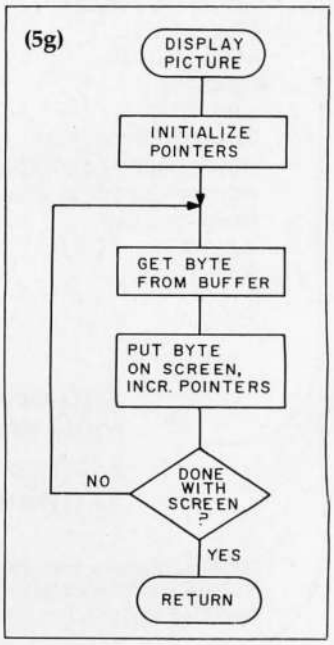
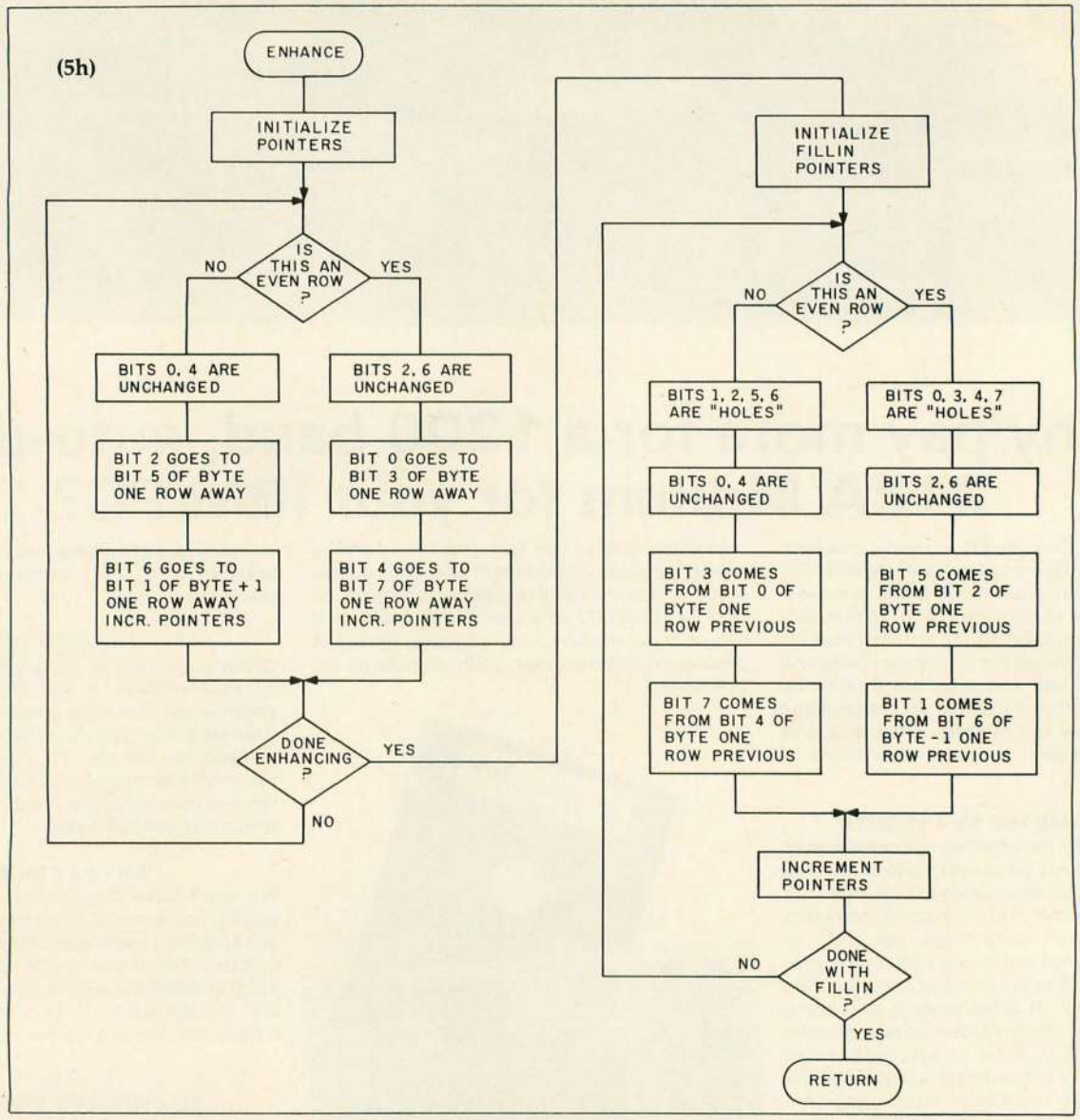


Figure continued on page 82

Figure 5 continued:



Text continued from page 76:

will be completely white. These situations may be remedied by increasing or decreasing the exposure time or changing the lens aperture. You may need to focus, also. Eventually, a clear picture will appear on the video screen when the lens is properly adjusted.

The gray-scale portion of the program demonstrates what can be done with just a little bit of software enhancement, permitting you to create images with 14 intermediate levels of brightness (plus extreme dark and bright) and display them on the Apple's hi-res screen. The image of an automobile shown in photo 1 is an example.

The technique used to display the gray-scale pictures on the Apple II Plus and IIe computers is known as *ordered dithering*, in which half-tone values are constructed from multiple binary black or white images. The process requires the Micro D-Cam system to take 15 exposures of the same subject, each lasting a little longer than the previous one. (This normally takes only several seconds.) After each exposure is taken, every pixel in it is checked. If the pixel is on (showing a 1 value corresponding to brightness above that exposure's threshold), a counter location corresponding to that pixel is incremented. At the end of 15 passes, this process yields a table of values, each value describing the relative intensity of its corresponding pixel. For example, if a pixel's final value is 15, that pixel should be displayed maximally bright; if a pixel's value is 8, the pixel deserves a shade of gray halfway between the black and white extremes.

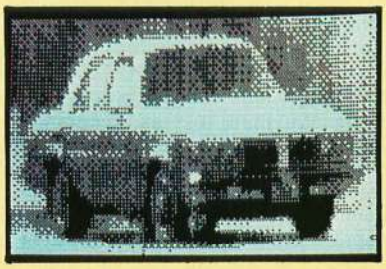


Photo 1: *The Micro D-Cam was aimed at a car parked outside. The dithered digital gray-scale image shown here is displayed by an Apple II Plus.*

Once the pixel-intensity table has been constructed, a 4 by 4 dither

Steven A. Ciarcia
Consulting Editor

BYTE
A McGraw-Hill Publication
70 Main Street
Peterborough, New Hampshire 03458
Telephone 603/924-9281

(2a)

(2b)

Photo 2: *A corner of the author's business card (2a) was easily reproduced by the Micro D-Cam (2b) because the printing represents only two levels of brightness. One potential use of the image camera is in optical character recognition.*

matrix is used to assign a display value (an array of binary pixels) to each screen position. In this software, the matrix is as follows:

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

Then, one pixel at a time, the values in the table of final image magnitudes are compared to the array-element values in the matrix. If the image value for the pixel is greater than the element's value, that array element is turned on. If the intensity value is 0, none of the matrix elements are displayed bright; if the value is 8, elements 0 through 8 are displayed bright; and if the value is 15, all the elements become bright. In this manner, 15 levels of luminance may be represented but at a certain loss of spatial definition. The process is repeated across the entire screen until each screen position has a value assigned to it.

It would definitely be possible to use different-size dithering matrices, with certain trade-offs. For example, a 2 by 2 matrix would yield only 5 levels of gray but would have much finer spatial definition, while an 8 by 8 matrix would yield 64 levels of gray but with much loss of spacial definition.

The GREY16 program overcomes many of the limitations associated with binary optical sensors. While black print on white paper (like my business card, shown in photo 2) is easily viewed by the Micro D-Cam with no enhancement, we don't live in a pure-black-and-white world, and three-dimensional objects need shading to be recognized on a two-dimensional video display.

This is most easily demonstrated with a series of photos of a pair of dice. Photo 3a shows the color and lighting conditions of our sample object. If we use the Micro D-Cam without gray scale, we obtain the binary picture in photo 3b. (This slightly vague yet quite representative picture of the dice would probably be usable in robotics or some recognition applications.)

For a more representative picture, we can invoke the G command in the GREY16 program to produce photo 3c. There is now no question of what the subject is or what value is shown on the dice. If the image were reproduced on a computer capable of displaying half-tones, it would look much more like photo 3a than this dithered Apple II Plus display.

(3a)

(3b)

(3c)

Photo 3: *A pair of dice (3a) was scanned by the Micro D-Cam. When only two levels of luminance are recorded and sent to the computer's display, the result is the output shown in photo 3b. When multiple gray-scale exposures and ordered dithering are invoked, the more easily recognizable output of photo 3c appears.*

Closing Observations

Two articles have not been enough to describe all the capabilities of the Micro D-Cam. If I had more time, I'd try some experiments using different lenses and filters. Theoretically, if three exposures were taken through red, green, and blue filters, we should be able to create a color image.

One interesting fact I did observe is that the IS32, like most silicon-based image sensors, is infrared-sensitive. My test was somewhat unscientific, and I have no precise data on the Optic RAM's spectral sensitivity. I merely lighted the subject with some infrared light-emitting diodes, but it was clearly seen by the Micro D-Cam even in visible-light darkness.

This mild success leads me to consider related experiments. Don't count on it, but in a few months you just might be reading about some sort of character-recognition wand I've built using an Optic RAM. In the meantime, if you find any other dynamic RAM chips that are suitable in this application or wish to show me a character-recognition program of your own, please write and let me know.

Next Month:

Communicating with their fellow humans can be a problem for people who cannot speak. We'll look at a way digital electronics can be harnessed to remedy this difficulty.■

Editor's Note: *For a review of a similar assembled product, see page 316.*

References

1. Ciarcia, Steve. "Analog Interfacing in the Real World." BYTE, January 1982, page 72.
2. Ciarcia, Steve. "Build the Micro D-Cam Solid-State Video Camera, Part 1: The IS32 Optic RAM and the Micro D-Cam Hardware." BYTE, September 1983, page 20.
3. Crow, Franklin C. "Three-Dimensional Computer Graphics." Part 1, BYTE, March 1981, page 54; Part 2, April 1981, page 290.
4. Grob, Bernard. *Basic Television: Principles and Servicing*, 4th ed. New York: McGraw-Hill, 1975.
5. Newman, William M. and Robert F. Sproull. *Principles of Interactive Computer Graphics*, 2nd ed. New York: McGraw-Hill, 1979.
6. Tomas, Joe. "Hardware Review: Dithertizer II." BYTE, February 1982, page 219.
7. Walker, Terry, Harry Garland, and Roger Melen. "Build Cyclops: First All Solid-State TV Camera for Experimenters." *Popular Electronics*, February 1975, page 27.
8. Williams, Thomas. "Digital Storage of Images." BYTE, November 1980, page 220.

The following items are available from

The Micromint Inc.
561 Willow Ave.
Cedarhurst, NY 11596

(800) 645-3479 for orders
(516) 374-6793 for information

1. *Complete Micro D-Cam unit including interface card, extension cable, IS32 Optic RAM, lens, remote housing, operator's manual, and utility software. Specify Apple II (II Plus or IIe) or IBM Personal Computer.*
   Assembled and tested.....$295
2. *Same as item 1 except in kit form. Specify Apple II or IBM PC.*
   Complete kit.....$260
3. *IC32 Optic RAM sold separately.*
   Each.....$42
4. *Serial-interface (RS-232C) Micro D-Cam for general use. Software listings for several different computers to be available soon. Call for price and delivery.*

Please add $4 shipping and insurance in continental United States, $20 overseas. New York residents, please include 7 percent sales tax.

Editor's Note: *Steve often refers to previous Circuit Cellar articles as reference material for each month's current article. Many of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.*

Ciarcia's Circuit Cellar, Volume I *covers articles that appeared in BYTE from September 1977 through November 1978.* Ciarcia's Circuit Cellar, Volume II *contains articles from December 1978 through June 1980.* Ciarcia's Circuit Cellar, Volume III *contains articles from July 1980 through December 1981.*

Special thanks to Carl Baker and Jim Herrud of Micron Technology Inc. for their contributions to this project.

To receive a complete list of Ciarcia's Circuit Cellar project kits available, circle 100 on the reader service inquiry card at the back of the magazine.

Steve Ciarcia (POB 582, Glastonbury, CT 06033) is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, product development, and marketing. In addition to writing for BYTE, he has published several books.