STATIC ROM

Build your own EPROM emulator for less than $60.00!

FRED EADY

THIS ARTICLE IS DEDICATED TO ALL electronics experimenters who have discovered the microcontroller. Developing a new microcontroller-based project is rewarding, but it can be difficult even with the aid of a microcontroller simulator. After all, you can't flash an LED or write to an LCD with a simulator because programs don't energize relays—hardware under program control does. Our inexpensive Static-ROM project is designed to be an alternative to the expensive professional microcontroller emulation systems. The Static-ROM in combination with an assembled 8048 board (discussed in greater detail later on) is the perfect learning tool for the beginner, and the perfect debugging tool for the experienced microcontroller user.

Designing projects that use EPROM (Erasable Programmable Read Only Memory) technology without reliable and helpful hardware tools can be an iterative and painful process. Each time you change the firmware embedded in the EPROM for your project, you must remove, erase, reprogram, and remount the EPROM. Even if you have dozens of EPROM's, and cycle through them as you change your firmware, it is still very time consuming. An alternative to development with EPROM's is the use of battery-backed RAM modules. But they are expensive, require special programming tools and techniques, and you still have to remove and remount them. Wouldn't it be nice to be able to emulate the EPROM in your project and eliminate the vicious erase/program cycle? Wouldn't it be nice to be able to build your own EPROM emulator in an evening for less than $60.00?

Our Static ROM project is named for its ability to emulate ROM (Read-Only Memory) with static RAM (Random-Access Memory). It plugs in directly where the EPROM would go, and eliminates the hassle of EPROM swapping. An average program change and load to Static-ROM takes less than a minute; it takes a minimum of 15 minutes to erase an EPROM!

Static ROM can be made to emulate the EPROM family from 2716 to 27256 simply by switching EPROM emulator cables. In addition, Static-ROM provides an automatic processor-reset pulse (active-high or active-low) after your program download that can be used to restart your target processor.

Operation

The Static-ROM is built around the PIC16C55 CMOS microcontroller (IC3 in Fig. 1). The 28-pin PIC16C55 provides one 4-bit and two 8-bit bi-directional I/O ports, as well as a timer/counter input, clock input, and clear input. The PIC16C55 contains 512 bytes of EPROM and 32 bytes of RAM. That doesn't seem to be much, but good things come in small packages. The EPROM bus is 12 bits wide, while the RAM bus is standard 8 bits wide. The 12-bit EPROM bus allows the use of a simple and powerful instruction set with emphasis on high-speed bit, byte, and register operations. There are only 33 12-bit instructions for the PIC16C55. With a 4-MHz clock, each instruction is executed in 1 microsecond. Its beauty lies in the power and simplicity of the PIC16C55 instruction set, as it takes less than 60 instructions to implement the entire algorithm for Static ROM.

The algorithm programmed into IC3 reads the STROBE line of parallel port 1 or 2, responds on the BUSY line of the respective parallel port, and supplies control line signals and sequential address information to the 43256 static RAM, IC4. Terminal software controls the selected parallel port, and communicates with the PIC16C55 and the user while routing binary user-written data to IC4. The PIC16C55, as implemented in the Static-ROM, is essentially an intelligent 15-bit up-counter triggered by the parallel port STROBE line and synchronized with the parallel port BUSY and END lines. The END input on the PIC16C55 serves as an end-of-download indicator, and also supplies the reset pulse for the target system's processor. Resistor SIP's R4 and R5 perform the necessary pull-up functions for the PIC16C55 output lines.

Static RAM chip IC4 (a 43256) holds all of the downloaded user-generated binary EPROM image data. If your application does not use raw binary data, Static-ROM does not care—it downloads whatever the terminal program sends. Therefore, special tables and unique characters can also be downloaded and used during emulation. The 43256 has the equivalent storage capacity (32K) of the 27256 EPROM. Only the amount of storage necessary to emulate a particular EPROM is used during actual emulation. In the program-download mode, the address lines of IC4 are controlled by the PIC16C55. In the EPROM-emulation mode, the 43256 is con-

IF, it is easier to select only the desired frequency range. At very low frequencies, where 1/*f* noise is likey to cause problems, it might be desirable to chop or upconvert that low-frequency signal to a higher band, especially if the signal source can be modulated.

A second major consideration is the selection of the first-stage active device, which usually depends on the frequency that the device must operate at and desired input impedance. For simple projects, the obvious choices are bipolar transistors or JFET's. For special projects such laboratory equipment, the designer might wish to use more exotic (and expensive) devices such as Josephson junctions, superconducting quantum interference devices (SQUID's), or tunnel-diode parametric amplifiers, all of which offer very low-noise inputs.

The crucial factor in bipolar transistor or FET selection is input impedance; FET's are inherently high-impedance devices, while bipolar transistors are lower impedance devices. Bipolars are favored for impedances in the 100-ohm to 1 megohm range, JFET's are satisfactory above 1K, while MOSFET's are the best choices for impedances above 1 megohm. If the input conditions are not established or subject to variation, as in an oscilloscope input, a JFET is preferable. JFET's offer much lower noise current than bipolar transistors, but their noise voltage is only slightly lower. Except when they must operate at very high frequencies, MOSFET's are at a disadvantage because of their large 1/*f* noise.

Once a device type is selected, the next step is to choose a specific device. This can be done by looking at the noise specifications on the data sheets. For best noise performance, it is advisable to choose a device that is operating in its midrange. Also, high-gain devices generally exhibit lower noise.

It is important that signal impedance be matched. Compare the source impedance (a function of frequency) vs. the device's noise current and voltage at the operating frequency, input impedance, and operating point. If the source impedance is less than a few hundred ohms, transformer coupling should be considered. Remember that if two FET's are in parallel, the effective input impedance will be halved. This doubles transconductance and noise current noise while halving the noise voltage.

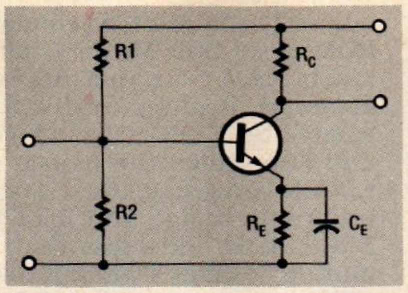


FIG. 9—COMMON-EMITTER AMPLIFIER with traditional biasing.

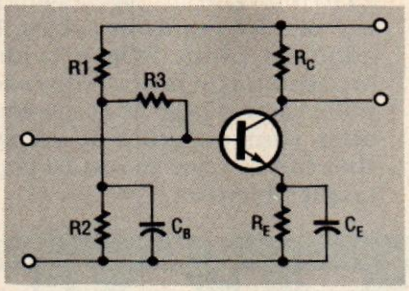


FIG. 10—COMMON-EMITTER amplifier with low-noise biasing. Noise from resistors R1 and R2 is shunted to ground by the capacitor.

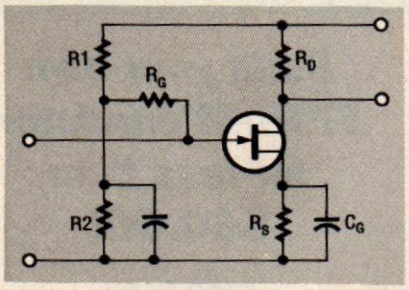


FIG. 11—JFET AMPLIFIER biased for low noise.

Suprisingly, feedback has little effect on noise performance. Except for the possible introduction of noise from the feedback resistors, the addition of feedback does not affect the S/N ratio. Also, the amplifier configuration—common base, common emitter, or common collector—does not significantly affect device performance. This gives you the freedom to choose the amplifier configuration that best matches the signal source impedance.

The next step is biasing. Figure 9 shows a typical common-emitter amplifier stage. For a high-gain stage, R_C is unlikely to contribute noise, and noise produced by R_E is shunted to ground by C_E. Resistors R1 and R2 bias the base. For minimizing amplifier noise, the values fo those resistors should be as small as possible.

Because these resistors have significant voltage across them, resistor excess noise must be considered. A tradeoff might be necessary. If calculations indicate that the excess noise is a problem, the circuit shown in Fig. 10 can be used. Noise from R1 and R2 is shunted by C_B. Only R3 contributes noise, but because the voltage drop across this resistor is low, excess noise does not present a problem. The analysis for a common-collector amplifier stage is similar to the common-emitter except that in this case the emitter resistor should be as large as possible to maximize the gain.

JFET biasing is illustrated in Fig. 11. Because of the low FET leakage current, R_G can be quite large so that it does not contribute much noise. Because a JFET's channel effectively isolates it from the gate, R_S and R_D are decoupled from the input. Therefore, noise from those resistors appears in the second stage, after the input signal has been amplified by the FET.

Once the input stage is designed, be sure that external noise is excluded. This means ensuring that the input signal source and input stage are well shielded and noise from the power supply is filtered out. Low-inductance capacitors can bypass power supply noise effectively. For more demanding applications, it might be necessary to use a separate modular power supply. In extreme cases, the isolation of a battery might be required. **R-E**

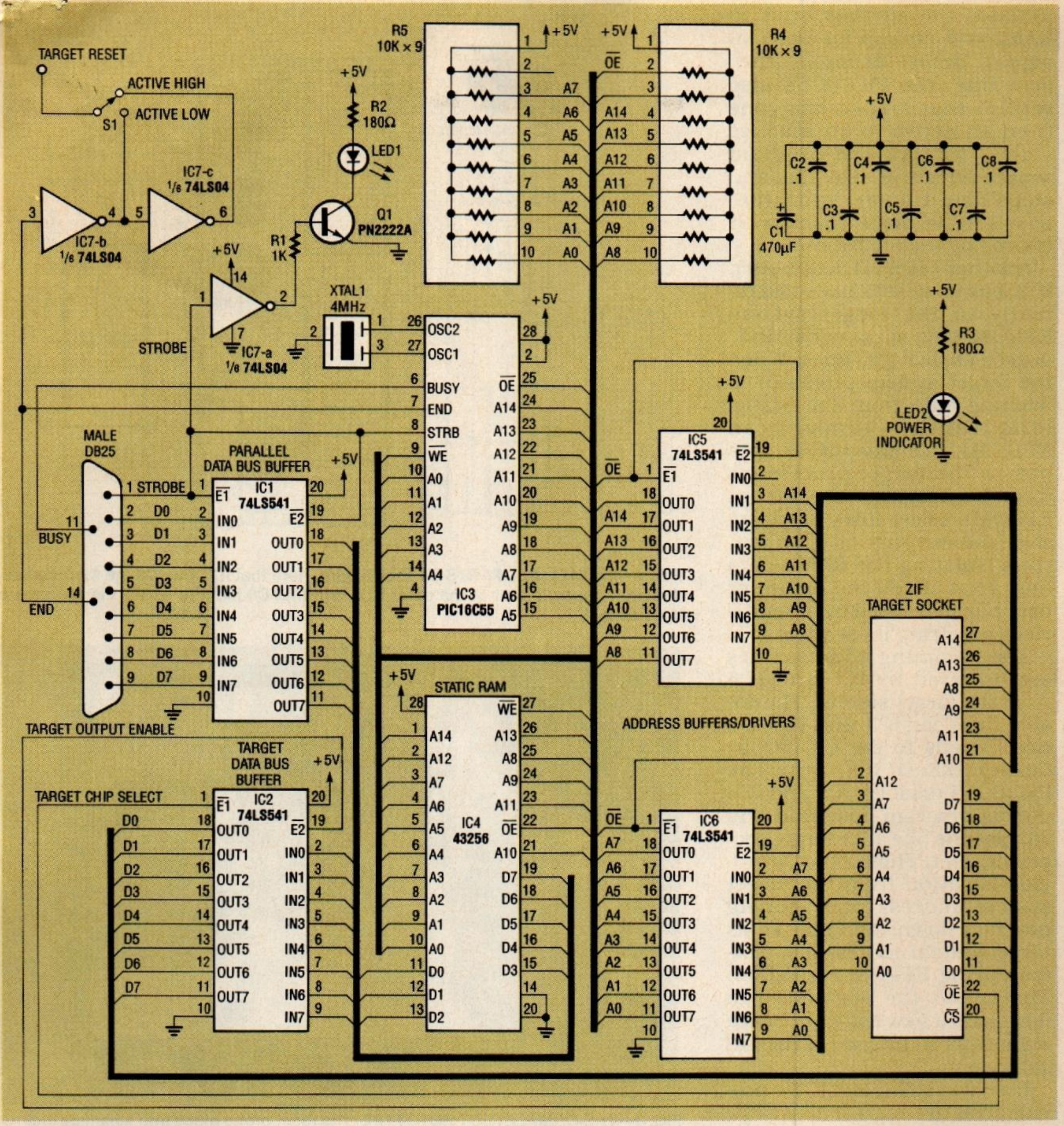


FIG. 1—THE STATIC-ROM is built around IC3, a PIC16C55 CMOS microcontroller that has one 4-bit, and two 8-bit bi-directional I/O ports, 512 bytes of EPROM, and 32 bytes of RAM.

nected via address and data buffers to the target system exactly as an EPROM would be. The target system does not know the difference.

While the PIC16C55 is generating address information for the static RAM, the static RAM address and data lines must be isolated from the target system's address and data lines to eliminate any conflict, and the static RAM data lines must be gated to the data lines of the parallel port. IC1, a 74LS541, provides the gate for incoming data from the parallel port. The chip passes data to the 43256 static RAM when the STROBE line is active (TTL low). When the STROBE line is inactive, IC1 isolates the parallel port data bus from the static RAM data bus.

When in the EPROM-emulation mode, the STROBE line is in the inactive state, which allows the static RAM data bus to be used exclusively by the target system. During the download cycle, the PIC16C55 emits a BUSY signal to the parallel port, and generates the WRITE ENABLE ($\overline{WE}$) signal required to store data in the 43256. During the BUSY time, the PIC16C55 generates the $\overline{WE}$ signal that writes the incoming data to RAM, increments the address counter, updates the address lines to RAM, and checks for the END signal. Before dropping the BUSY line, the PIC16C55 also verifies that STROBE has gone from an active to an inactive state. The terminal program senses an active BUSY line and stops data transfer until the BUSY line is made inactive or cleared by the PIC16C55.

Another 74LS541 octal buffer, IC2, has its ENABLE lines tied directly to the target system EPROM CHIP SELECT $(\overline{CS})$ and OUTPUT ENABLE $(\overline{OE})$ lines. When the target system processor is fetching data from the Static-ROM (which it thinks is an EPROM), IC2 gates IC4's data bus to the target processor's data bus. When the target EPROM's select lines are inactive, the outputs of IC2 float, thus isolating the RAM's data bus. That allows the processor data bus to be used by other devices requiring its services.

The incoming address lines are buffered by IC5 and IC6 from the target system. The enable lines for IC5 and IC6 are tied directly to the PIC16C55 OUTPUT ENABLE $(\overline{OE})$ pin. The PIC16C55 disables IC5 and IC6 during program download and enables the line after target processor reset. Those two IC's provide isolation from the target system address lines during program download, and act as target-system address-line drivers during EPROM emulation. That allows the Static-ROM to download a new software image regardless of the state of the target system.

Inverters IC7-b and IC7-c provide an active-high or active-low RESET pulse in synchronization with the END input from the parallel port. Gate IC7-a, driven by the parallel port STROBE signal, drives the base of switching transistor Q1 via R1 to indicate program download activity with LED1 and R2. Upon sensing an active END signal, the PIC16C55 resets its internal address counter to zero, refreshes the address bus to RAM, floats the address lines to RAM, enables the address buffers (IC5 and IC6), and waits for the next active STROBE signal from the parallel port. Upon receiving an active STROBE signal, the PIC16C55 disables IC5 and IC6, which isolates the Static-ROM address bus from the target. The processor then proceeds with loading the incoming data from the parallel port into RAM. Once the download is complete, the cycle repeats.

Static-ROM doesn't require personality modules, jumpers, and software to define the EPROM being emulated. Instead, each EPROM has a particular cable wiring scheme that allows the user to switch EPROM's simply by installing the appropriate cable between the Static-ROM target socket and the target system's EPROM socket. An advantage to this scheme is that you need only make the cables you will actually use.

Capacitors C1 through C8 filter and bypass the 5-volt DC power bus. LED2 along with re-

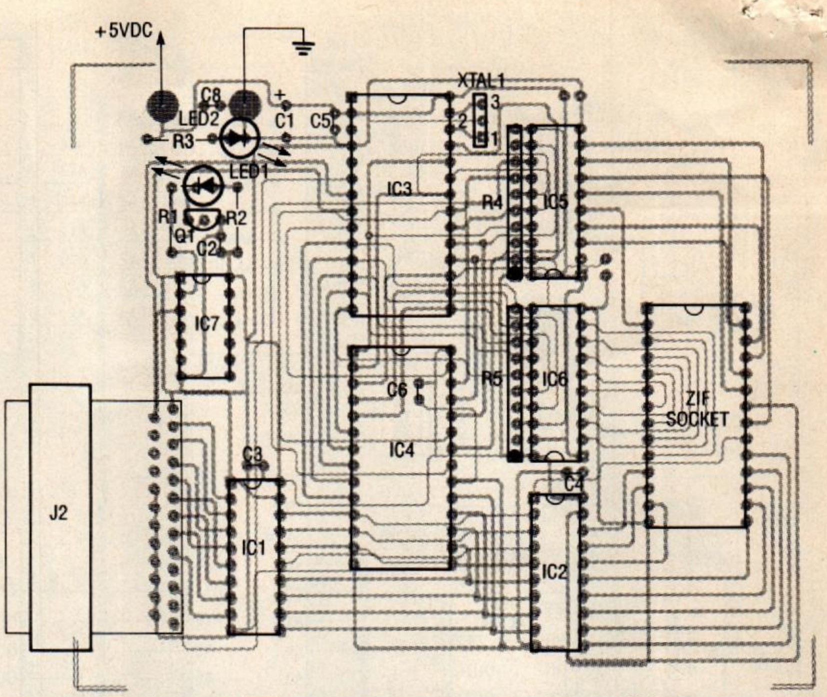


FIG. 2—INSTALL ALL PARTS as shown here. Note that IC5 and IC6 are mounted in a direction opposite that of the other IC's, and that C6 mounts within the confines of IC4's socket.

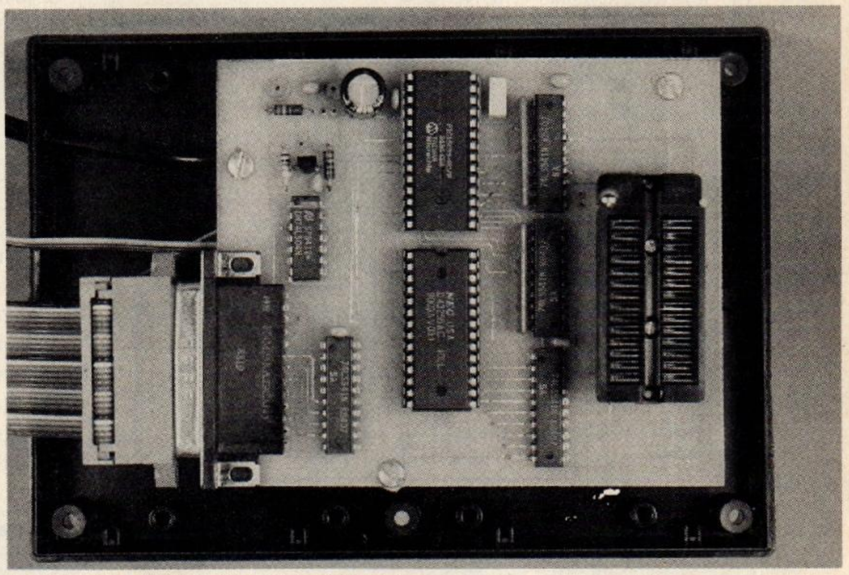
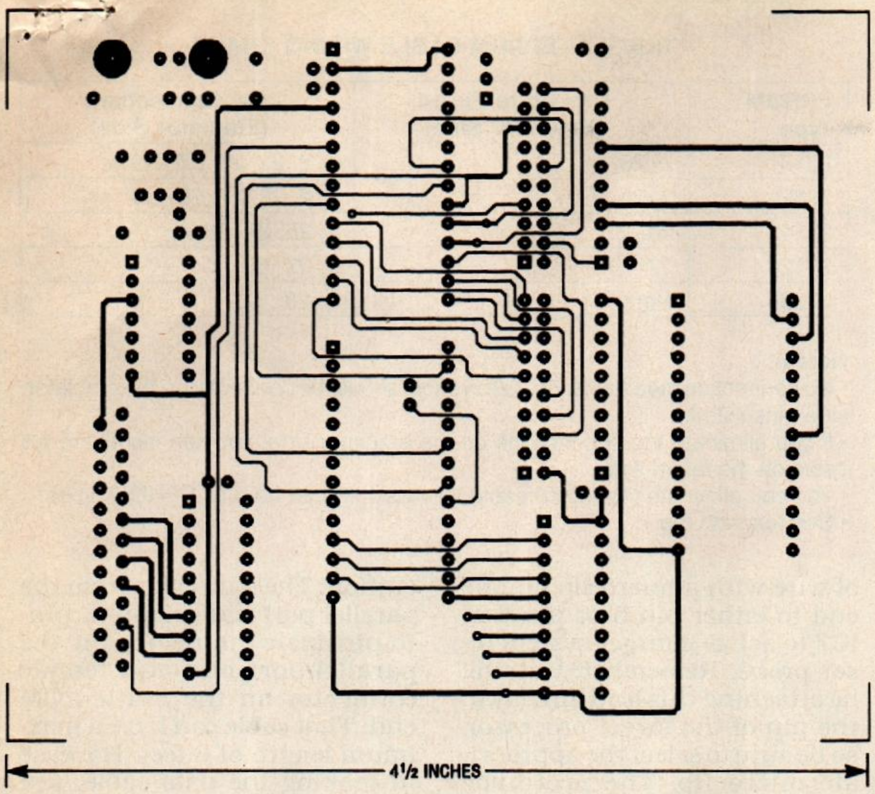
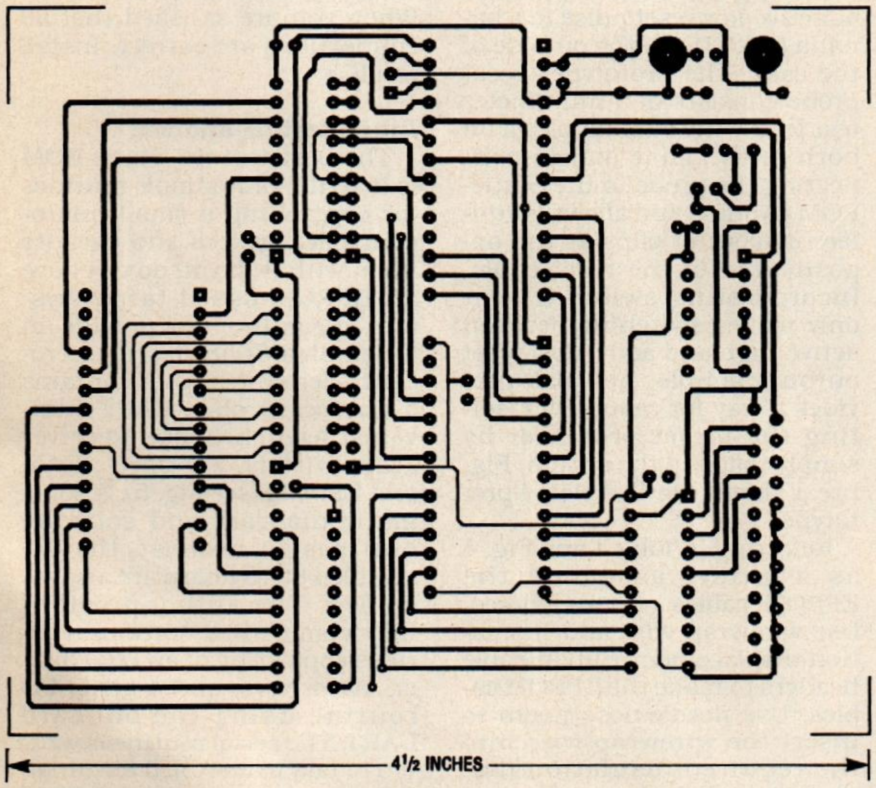


FIG. 3—THE COMPLETED PROTOTYPE. The reset probe plugs into a banana jack, J1, on the outside of the case.





sistor R3 serve as a POWER-ON indicator. An external 5-volt DC regulated power supply is required to power the Static ROM.

The software for the Static-ROM consists of a terminal program that transports user-generated data, and controls the selected parallel port. It has an intelligent address generation algorithm that resides within the PIC16C55. If you own a PIC16C55 programmer, the PIC16C55 source listing is available, so you can program your own. Source listings of the software for both the PIC16C55 and terminal program are available on the RE-BBS or from the address in the Parts List.

Construction

A PC board is recommended, but not necessary. If you choose to make one, use the supplied foil patterns; otherwise you can obtain one from the address in the the Parts List. Placement of the DB25 parallel-port connector is critical, so follow the

PARTS LIST

Resistors
R1—1000 ohms
R2, R3—180 ohms
R4, R5—10,000 ohms × 9, SIP
Capacitors
C1—470 µF, 16 volts, electrolytic
C2–C8—0.1 µF, Mylar
Semiconductors
LED1, LED2—light-emitting diode, any color
Q1—PN2222A NPN transistor
IC1, IC2, IC5, IC6—74LS541 octal buffer/line driver
IC3—PIC16C55 CMOS microcontroller (programmed)
IC4—43256 static RAM
IC7—74LS04 hex inverter
Other components
XTAL1—4-MHz ceramic oscillator
S1—SPDT switch
J1—banana jack
Miscellaneous: 5-volt DC power supply (500 mA), 28-pin ZIF socket (optional, for the target socket), IC sockets, 25-pin right-angle D shell connector, EPROM cables, serial cable, PC board, wire, solder, mounting hardware, etc.
Note: The following items are available from Fred Eady, PO box 541222, Merritt Island, FL 32954:
- **A complete kit of parts including PC board, 25-pin connector, SPDT switch, and IC sockets (not including 8048 target board, power supply, ZIF socket, and cables)—59.95 + 5.00 S&H**
- **Assembled 8048 microcontroller target board and software routines on diskette—$20.00 + 5.00 S&H**
- **Programmed PIC16C55—$25.00 + 5.00 S&H**
- **PC board only—$15.00 + 5.00 S&H**
- **Static-ROM software on diskette—$5.00 postpaid**

Check or money orders only. For technical assistance and inquires call 407-454-9905

component layout if you choose to handwire the Static-ROM.

It's a good idea to socket all of the IC's. Install all IC sockets as shown in the parts-placement diagram of Fig. 2, paying particular attention to the pin-1 positions of IC5 and IC6, which are mounted in the opposite direction from the other IC's. Capacitor C6 mounts within the confines of the socket for IC4. The dot on SIP resistors R4 and R5 indicates pin 1, as do the square pads on the printed circuit board. The Static-ROM does not require a case, but if you decide to use one, remember to mount the LED's and switches in the appropriate locations.

At this point you should have all of the components except the IC's and the DB25 shell connector mounted on the printed-circuit board. Apply power (the POWER LED should illuminate) and check for power and ground on all of the IC's. If the specified voltages are not present, recheck your work. When you are satisfied with the voltages, install the DB25 connector.

Depending on your choice of target processor, solder a length of wire with a microclip on one end to either pin 5 or pin 6 of IC7 to act as a target-system reset probe. Remember, you will be attaching this lead directly to the pin of the target processor, so be sure to select the appropriate microclip. The prototype was built with SPDT switch S1 supplying either an active-high or active-low reset pulse to a banana jack, J1, on the outside of the case. The prototype's reset probe consists of a multimeter test lead with banana plugs on both ends. That makes connecting the probe to the Static-ROM painless and allows for different types of clips at the opposite end of the reset cable. Incorporating switch S1 not only makes switching between active-high and active-low reset outputs simple, but also provides a way for manually resetting the target processor by simply toggling the switch. Figure 3 shows the completed prototype.

Referring to Table 1 and Fig. 4 as a guide, assemble the EPROM cables of your choice. Use wirewrap wire and insulation-displacement ribbon-cable headers to make the EPROM cables. Use needle-nose pliers to insert the wirewrap wire into the required insulation-displacement header pins, and then insert the ribbon cable on top of the wirewrap jumpers, and then complete the assembly of the header. Maximum EPROM cable length should not exceed 36 inches for reliable operation. The data cable from the parallel port has a 25-pin, pin-to-pin male connector on the parallel-port end and female connector on the Static-ROM end. That cable can have a maximum length of 6 feet. For ease in making the data cable, use insulation-displacement DB25 shell connectors on both ends. When you are satisfied that all connections are correct, install the IC's.

TABLE 1—EPROM CABLE WIRING CHART

EPROM Type	Ground to Pin 14 (Emulator Side)	No Connection (Emulator Side)
2716	2, 23, 26, 27	1, 2, 23, 26, 27, 28
2732	2, 26, 27	1, 2, 26, 27, 28
2764	26, 27	1, 26, 27, 28
27128	27	1, 27, 28
27256	None	1, 28

Notes:
- No connection means that no ribbon cable should be connected to the emulator-side pins listed.
- If you eliminate the connections on the emulator side, you can install the full cable on the target side.
- You can eliminate the ribbon cable for pins 1 and 28 from all EPROM types.
- See text and Fig. 4

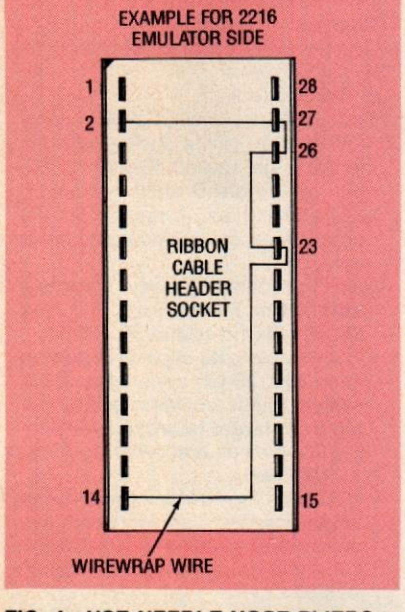


FIG. 4—USE NEEDLE-NOSE PLIERS to insert the wirewrap wire into the insulation-displacement header pins, and then insert the ribbon cable on top of the wirewrap jumpers (see text). Table 1 shows the connections for different EPROM's.

Initial testing and use

The best test for Static-ROM is to write some simple routines for controlling a small microcontroller system and execute them with it. If you do not have an EPROM-based target system, the author will provide an assembled 8048-based microcontroller system that contains a microcontroller, UART (Universal Asynchronous Receiver Transmitter), I/O ports, RAM and EPROM (see Fig. 5). A schematic diagram and software routines to exercise the assembled 8048 board are also included. The author-provided fully commented software routines consisting of an LCD driver, I/O drivers, and a serial I/O routine using the onboard UART. The serial routine is written to talk to an ASCII terminal. A feature of the assembled 8048 board is that you can replace the 8048 and EPROM on the assembled board with a programmed 8748. This feature permits you to eliminate the EPROM if you decide to dedicate

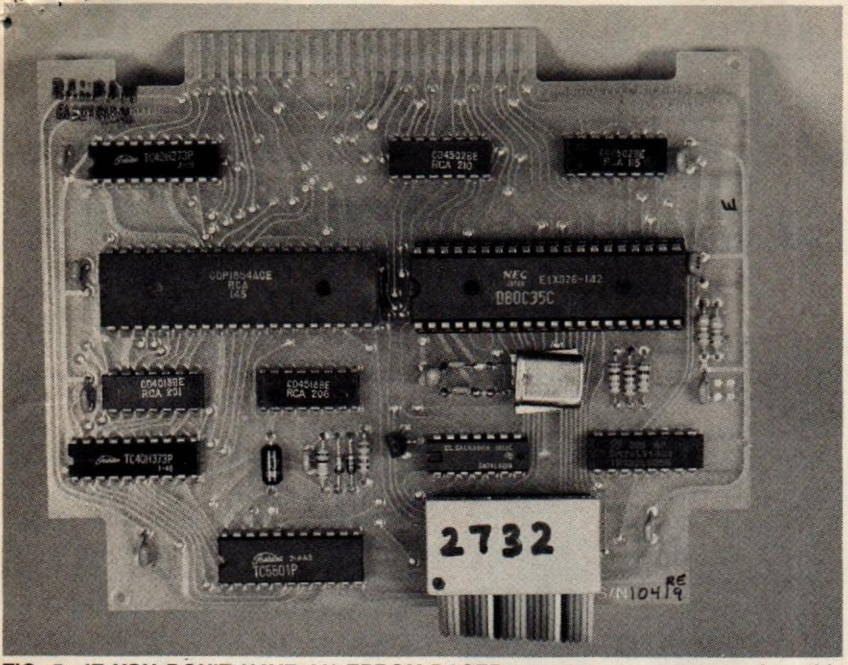


FIG. 5—IF YOU DON'T HAVE AN EPROM-BASED target system, you can buy this assembled 8048-based microcontroller system consisting of a microcontroller, UART, I/O ports, RAM and EPROM (see the Parts List).

the board to a specific application and do not wish to use a separate EPROM. See the parts list for ordering details.

When you have obtained a working target system, connect the 25-pin DB25 male-to-female cable, with the Static-ROM to either parallel port and apply power to the Static-ROM. If the circuit is functioning correctly, the POWER LED should be illuminated, and the LOAD LED can be off or on, depending upon the initial state of the parallel port STROBE line.

Next, connect the EPROM cable to your selected target system. Note that the author-supplied 8048 system uses the 2732 EPROM cable. Select either active-high or active-low reset, depending on your processor. The 8048-based system offered by the author uses active-low reset. Connect the reset probe to the reset pin of your target processor. When you start the Static-ROM terminal program, you will be prompted to select the parallel port that is connected to the Static-ROM. If the LOAD LED was on, it should now be off. Enter the name of the binary object file you want to download to Static-ROM.

After pressing the enter key, you should see the LOAD LED illuminate and a "sending" message with a byte count on the terminal display. The terminal program should indicate that the download has completed, and the target system processor has been reset. You should see immediately the downloaded program running on your target system. Toggle the reset select switch to reset your target system manually or run the Static-ROM terminal program again, and download and run another binary file.

The Static-ROM is designed as a debugging tool, not an EPROM eliminator or substitute. Disconnecting Static-ROM from the parallel port while emulating an EPROM will terminate the emulation if a randomly generated active level is presented to the PIC16C55 STROBE input. Recall that the PIC16C55 always wants to restart on an active STROBE from the parallel port while in EPROM emulation mode and, if an inadvertent strobe is encountered, the algorithm will cease looking for download data that is not forthcoming.

As you can see, you can transfer between your assembler and the terminal program quickly, testing your code until you obtain the desired results. Ω

Earn Your B.S. Degree in ELECTRONICS or COMPUTERS

By Studying at Home

Grantham College of Engineering, now in our 43rd year, is highly experienced in "distance education"—teaching by correspondence—through printed materials, computer materials, fax, and phone.

No commuting to class. Study at your own pace, while continuing on your present job. Learn from easy-to-understand but complete and thorough lesson materials, with additional help from our instructors.

Our Computer B.S. Degree Program includes courses in BASIC, PASCAL and C languages — as well as Assembly Language, MS DOS, CADD, Robotics, and much more.

Our Electronics B.S. Degree Program includes courses in Solid-State Circuit Analysis and Design, Control Systems, Analog/Digital Communications, Microwave Engr, and much more.

An important part of being prepared to *move up* is holding the right college degree, and the absolutely necessary part is knowing your field. Grantham can help you both ways—to learn more and to earn your degree in the process.

Write or phone for our free catalog. Toll free, 1-800-955-2527, or see mailing address below.

•

Accredited by the Accrediting Commission of the National Home Study Council

•

GRANTHAM College of Engineering
Grantham College Road
Slidell, LA 70460

HARDWARE HACKER

Digital photo imaging, a pair of new GPS books, magnetic field resources, fluxgate compass papers, and error diffusion techniques.

DON LANCASTER

Several readers have reported problems when logging on to my *GEnie* PSRT board. The trick is simple: Be sure to enter the HHH *immediately* after your communication software verifies a connection. You will then be prompted to enter an account number, and can proceed from there.

I have looked further at one of our "free energy" resources from a few columns back. *Rexearch* is one well-done labor-of-love operation. What Rex has done is gather together his lifetime collection of weird science, junk science, pseudoscience, and utterly off-the-wall papers, patents, and press coverage. His typical $6 packages might include full copies of several patents and an article or two. There are hundreds of topics. All of them strange and wondrous. A free master list is offered.

Yawn. Stupendous breakthroughs are a thankless task, but somebody has to do them...

Spotless halftones

I've recently been exploring ways to improve the quality of digital photos produced on laser, dot-matrix, or inkjet printers. By going to a new *spotless halftone* technique that does non-obvious stuff in non-obvious ways, print quality can be very much improved.

Most "experts" would agree that the "best" photo halftone you could do at a plain old 300-DPI resolution is something like 19 grays at 70 spots per inch or 15 grays at 85 spots per inch. That, at its very best, is worse than "Sunday funnies" or "auto shopper" tabloid print quality.

Instead, Fig. 1 shows a portion of a stock Lena digital photo, done with a brand new *spotless* 300 DPI halftone. This offers 65 grays, a grain that seems much finer than 106 spots per inch, and full com-

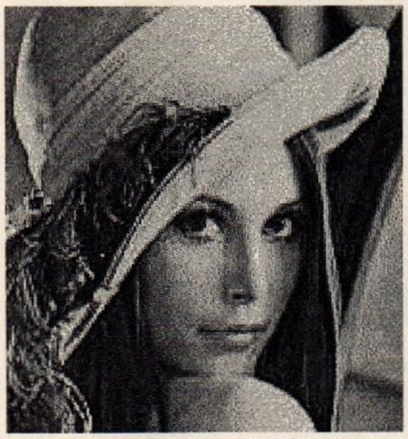


FIG. 1—A SPOTLESS HALFTONE reduced by about 50% from an unenhanced 300 DPI laser printed original. With care, this new technique offers significantly better image quality than traditional methods.

patibility with traditional ink. I'd dearly love to show you the 600-DPI version, so I'll show you where to get your own copy shortly.

The bottom line: You can now print your own "useful" photos at 300 DPI and "very good" photos at 600 DPI for top-quality book-on-demand publishing. They're often good enough to eliminate the need for high-resolution typesetters.

Traditional photography is an extremely flexible and forgiving process. You can raise or lower brightness by opening up or closing down a stop. Change the contrast by printing one paper grade higher or lower, or custom dodge out all your muddy grays and burn in your harsh highlights. Simple and easy.

Unfortunately, digital halftones lack that flexibility. There is absolutely no room for forgiveness. Let's look at some of the essential tools we now have to improve digital halftones. We'll also summarize them in Fig. 2.

Decent input data—"Garbage in, garbage out" is especially true for digital halftones. Unless you start off with your best possible image data, a disaster is a certainty. Each *pel*, or picture element, in the original should correspond *precisely* to *one* pel in the final image. Those pel values must be *linearly* coded to at least 64—and preferably 256—gray levels.

There's lots of tools available to do the job, such as *Adobe Photoshop*. If you are careful, photo enhancement can be done by using the PostScript in your printer as a general-purpose computing language.

Cropping—This is just throwing away outside areas that do not add to your image content. Always get the key theme or subject as obvious and as dominant as possible. But keep your aspect ratio at something sane, like 1:1 or 4:3 or 1.62:1. Photos at other aspect ratios can be visually jarring.

At any rate, the key point is to start off with the very best image you can. And then enhance it using every trick in the book.

Histogram equalization—Within limits, the more gray levels you can print, the better your results. If you have only a few grays available, you don't want to ignore some of them. That can only make matters worse. Too few grays give you a step effect with annoying visible jumps between the gray levels.

NEED HELP?

Phone or write your **Hardware Hacker** questions to:

Don Lancaster
Synergetics
Box 809-EN
Thatcher, AZ 85552

(602) 428-4073

For fast **PSRT** access, modem (800) 638-8369, then an HHH. Then XTX99005,SCRIPT.