Once upon a time, in a magic kingdom far away, there lived a truly wondrous automobile. It was the first "no excuses, no apologies" automobile ever available, and was very popular among the princes and the populace alike. Alas, this otherwise stupendous machine had a single and very serious flaw. A flaw so insane that it could only have been placed there by a demented and wicked witch.

For, you see, this wondrous automobile had a large pedal on the floor that was plainly marked "BRAKE." Drivers of this automobile expected and assumed that, when this "BRAKE" pedal was pressed, the automobile would be brought to a swift and safe stop, without harm to the driver, or to any of the passengers, or to any cargo.

But this was nought to be. Forsooth, the "BRAKE" pedal was really a magical pedal under a horrible spell. If the "BRAKE" pedal was pressed by itself, the automobile, the driver, and all contents got magically and instantly whisked back to the carriage house from whence the trip began.

If a driver was foolish enough to press the "BRAKE" pedal at the same time he turned on the windshield wipers, the wondrous automobile did, in fact, come to an immediate stop. But, alas and alack, the immediate stop was so sudden and so violent that it destroyed the driver, passengers, and all contents of the magical vehicle.

Well, not really destroyed. For, you see, all that really happened is that a pair of holes, five urfglogs in diameter, got neatly punched completely through the driver, any passengers, and all else that hap-penstance had placed inside the wondrous machine.

When the grand viziers were asked why the "BRAKE" pedal was not really a "BRAKE" pedal, but instead an evil and demonic device, they offered two reasons.

Some viziers said that the multitudinous makers of hood ornaments and glove compartment door hinges did not want the drivers bringing their vehicles to a quick and safe stop, since the hood ornaments and glove compartment door hinges could then—horror of unspeakable horrors—actually be inspected and possibly modified by the driver.

Others said that there must be some protection to keep the driver from inadvertently and unintentionally pressing the "BRAKE" pedal if he did not, in fact, really want to bring this wondrous automobile to a swift and safe stop. And indeed, a much older model of the same wondrous automobile, did have its "BRAKE" pedal situated where it could easily be mistaken for the horn ring.

Lo and behold, a certain driver of the wondrous automobile finally decided he had more than enough of this male bovine excreta. He pulled out the old "BRAKE" pedal by its roots and threw it away. Then, he replaced the magical "BRAKE" pedal with a real one that was able to swiftly and controllably bring this wondrous automobile to a safe and sure stop. Having done so, the driver grabbed the nearest handy princess, drove off into the sunset, and lived happily ever after.

Upgrading Apple IIe’s ROM Monitor

Here are fixes to change this model’s restrictive monitor to an absolute old reset one, giving users an opportunity to unleash the computer’s real power.

By Don Lancaster

A fable, you say? Only perhaps.

The wide-selling Apple IIe computer, likely the last Apple computer with reasonable expansion flexibili-ty, including socketed ICs, is not as tractable as adventurous users would like it to be. Its ROM monitor, where the system’s routines lie, has a fatal flaw of sorts. It’s not in the code itself, but in what a resident monitor should allow a user to do beyond what its cre-ators think you should do.

To get past the built-in restrictions imposed by the machine’s system monitor, so that you can do all sorts of wonderful things—emulate a “snapshot” card, replace the entire operating system with your own cus-tom code for dedicated data-ac quisition purposes, perform certain CAD/CAM programs, run special turnkey applications, etc.—I’ll show you how to design an improved mon-itor, like those in older Apples before the mass-market syndrome set in. With this ROM change, you’ll be in control, custom-modifying “App-lesloth” to your heart’s desire.

Fortunately, the IIe monitor is res-ident in chips that can be swapped easily for 64K EPROMs. Along the
way I'll show you how to burn in the new EPROMs (which cost only about $6 each), that will replace the original ones including putting in the necessary code. The improved monitor will do everything the original IIe monitor ROM does, as well as open new computing vistas for you. Moreover, there is no "hole blasting" done on a cold restart, which essentially destroyed pages of code on a reset, taking away marvelous tools such as stopping and inspecting. And if you keep your finger on the Open-Apple key for at least four seconds after a cold reboot, you will be dropped directly in the "old," more versatile monitor, awaiting your every machine-language command.

The code version covered here applies only to the "original" IIe ROMs, shipped in most IIe's up to late fall of 1984. Separate code is required for either the "new" IIe ROMs or for the IIc. (See box at end of article for details on getting code for versions.) You can tell you have an "original" IIe ROM if your scrolling appears disjointed and chunky.

I call this new monitor a KREBF monitor, named after the magic spell that "repairs willful damage" in Infocom's Enchanter adventure game. Actually, we won't repair any willful damage... we just won't do any in the first place. I'll show you a painless and fully automatic process for converting your own firmware into a form useful for EPROM programming. I will also give you the full new source code patch that will return absolute control of your IIe back where it rightly belonged in the first place. (This is also available ready-to-run on a companion diskette, as well.)

About The Firmware

The stock Apple IIe has 16K of ROM-resident firmware that sits between $C000 and $FFFF. As Fig. 1 shows, this firmware is held in two 64K read-only memories. One of these is called the CD memory and sits in main board location E8, while the other is called the EF memory and sits in board location E10.

The traditional monitor area to older Apple IIIs needed only the 8K space from $F800 to $FFFF. This area is still used as a monitor for the IIe. To provide for the many new IIe features, expansion hooks have been added to also allow use of the $C000-CFFF memory area.

Since $C000 to $CFFF is in the I/O (input/output) area, special soft switches are used to pick either "normal" I/O or "monitor" use of this address space. Thus, anything that wants use of the "new" IIe monitor area must first turn off the I/O space and then turn on the CD firmware ROM. When you are finished using the new monitor area, the CD ROM must be turned back off and the I/O space must be reactivated.

There are four soft switches involved. One pair handles only the memory area from $C300 to $C3FF and is used to make the 80-column firmware look like it is sitting in slot 3 of the I/O space. The second pair of soft switches is used to switch everything else. These are called the $CXX0 switches. The $C here can be a 1, 2, 4, 5, 6 or 7.

The top half of the CD memory and the bottom three-quarters of the EF memory hold the Applesoft firmware. This code is apparently unchanged from earlier models, in the "original" IIe firmware. The "new" IIe and IIc firmware made some changes to Applesloth, such as allowing lower-case commands.

Older Apples used 16K ROMs that were not quite compatible with industry standard 2716 EPROMs. One enable pin was active-high on the

---

Fig. 1. Apple IIe firmware is held in a pair of 8K × 8 ROMs. Standard 2764 EPROMs can be directly substituted for the supplied ROMs.
ROM, compared to an active-low state needed by a 2716 EPROM. While you could just change some jumpers around to do some short-sighted 2716 replacements, you really needed a small plug-in card to actually do the job right. Without the inverter, certain plug-in cards could cause memory contention and hang up the machine. Quite a few articles on adding EPROMs to older Apples ignored this key point, leaving you with a potential time bomb on your hands. Thus, swapping for 2716 EPROMs did not get done much on older Apples because of the hassles involved.

Very fortunately, the 64K ROM firmware used in the IIe is directly and exactly compatible with the standard 2764 EPROM. So, to customize things any way you like, you simply swap chips. The “CD” or “EF” chips in a IIe may be directly replaced by 2764 EPROMs. In case you’ve tuned into the microcomputer revolution late, a 2764 is a special “read-mostly” memory that you can custom program and reprogram yourself. You buy these for around six bucks. Erase any old memory contents by using a special ultra-violet lamp. You reprogram the memory to suit your application, using either a programming card for the Apple or a stand-alone EPROM programmer.

EPROM programmers are readily available at any hacker’s club if you do not already own one. We will shortly see a sneaky way to program the 2764 on older burners that may not be directly able to handle such a large EPROM.

To customize your IIe, all you have to do is burn one or two 2764 EPROMs and then swap them for the CD or EF memory they are to replace. That quick and that simple. There is one minor “gotcha” though: the 2764 EPROM used must be an Intel or Hitachi brand with 28 pins and an access time of 250 nanoseconds. Real 2764s always come in a 28-pin package, and are the only type of 2764 useable as a direct Apple monitor replacement. (Both Motorola and Texas Instruments have their own imitation versions of fake 2764’s that come in 24-pin packages.)

About 64K Eproms

Figure 2 shows the pinouts for a real 2764, along with some use details.

To analyze any memory chip, break the package leads down into four groups: supply, address, data, and control lines. Then analyze each group as follows:

- Only a single +5-volt supply and ground are needed for normal reading of the 2764 EPROM. Supply current is typically around 150 milliamperes in the read mode.
- The 2764 is a 64K memory that is organized as 8K by 8. This means that there are 65,536 bit locations that are programmable to 1 or 0. These bit locations are arranged into word bytes of 8 bits each. Thus, there are 8192 different 8-bit words.
- The address lines select which word is to be written to or read from. Thirteen address lines are needed, since $2^{13} = 8192$. To select a particular 8-bit word, the correct binary pattern of 1s and 0s is placed on the address lines. Internal address decoding inside the chip then selects the correct byte for reading or writing. Address lines always input from the microcomputer to the 2764.
- There are eight data lines, used to route the contents of the addressed byte to the microcomputer during a read. The same lines are used to send the data to be written into the EPROM during a write. Thus, the data lines input to the EPROM during a write, and output from the EPROM during a read.
- There are five possible activities an EPROM can be up to:
  1) Erase—clears the entire memory when exposed to strong UV light.
  2) Program—writes a single byte into memory.
  3) Verify—checks a byte just written during programming.
  4) Read—outputs a previously programmed data value.
  5) Standby—does nothing, allowing other data bus uses.

These activities are handled by four control lines, called VPP, PGM, OE, and CE.

To use an EPROM, you first have to program it. Before you program it, you have to erase anything previously stored in it. Erasure forces all of the data bits to 1s, and all bytes to $FF$s. This erasing is done with a special ultraviolet lamp.

The VPP (programming voltage) line provides normal +5 volts dc for standby and reading. To program, this line must be brought to +21 to +24 volts. Earlier EPROMs did not have this programming voltage on a separate pin, which complicated things. On some micro systems, it is a simple matter to in-socket program a 2764. Unfortunately, the stock Apple IIe does not have this capability. At any rate, leave VPP at +5 volts to do anything but program. Raise it to +24 volts to program.

Incidentally, most 2764s can use either a +21- or a +24-volt VPP programming voltage. Some older 2732As and a few 2764As demand a maximum of +21 volts while being programmed and will self-destruct on the traditional +24 volts. Check the data sheet for the exact brand you are using if there is any doubt.

The PGM (program) line is a normal logic control signal. This one is grounded to program and is set to +5 for everything else.

The OE (output enable) line is brought low for a read or a write. A high OE line disconnects the outputs of the EPROM, but leaves the chip addressed and fully powered.

The CE (chip enable) is used to turn on or off the entire EPROM. During a read, CE must be brought...
low. During a write, CE is held high until an address is selected and data is input. Then CE is brought low for exactly 50 milliseconds to blast the data value into the EPROM. CE is then returned high while the address changes for the next byte. A high CE also reduces the standby supply power needed.

Apple chose to use the OE line as an alternate chip-enable, since it is faster. They were not in the least worried about saving any supply power, since they already had problems meeting the minimum power supply current drain during the IIe redesign. Apple also permanently grounded the CE line and permanently held the PGM line at +5 volts, forcing any 2764 plugged into the main board into a read-only mode.

Thus, unless you want to do some major board carving, “in situ” programming of a 2764 EPROM in an Apple IIe is not normally feasible. Instead, you have to use an EPROM programmer or programming card and then swap out the chips. For more information on this and other memory chips, check into Don Lancaster’s Micro Cookbooks, SAMS #21828 and #21829.

### Programming A 2764 On An Older Burner

Many older EPROM burners cannot directly handle a 2764, and you won’t find too many newer models available yet that do. Some of the nasties involved in physically upgrading an older burner include going from a 24-pin to a 28-pin socket, providing that thirteenth address line, getting a PGM signal to the socket, and figuring out how to stuff 8K worth of data into the 4K I/O space normally available on an Apple II.

We can bypass all these hassles by adding a fairly simple and easy-to-use adapter to an older EPROM burner. This adapter will make the 2764 to be programmed look like a sequential pair of older 2732s. If you can can program a 2732, as most older EPROM burners can, you can easily program a 2764 with this adapter.

You will find a two-position slide switch on the adapter. (Caution: Do not use a toggle switch here; it works “backward.”) Put the switch in the low position to program the low 4K of your 2764 as if it is a 2732. Then flip the switch to the high position so you can program the high 4K of your 2764 as if it is a separate 2732.

Our adapter is intended for use with the MPC “ap-ep” EPROM burner peripheral card for the Apple II or IIe. Certain details might change for other burners.

Listed in the table are the parts you will need to fabricate your adapter. These parts should be readily available anywhere, although you must be certain to use the type of premium machine-contact DIP socket that can safely be plugged into one another.

A schematic of the adapter is shown in Fig. 3. Note that this adapter and the tools required to build it...
are needed only if you are having trouble burning a 2764 EPROM on an older EPROM burner. All you need to do the actual upgrade on your IIE is a programmed 2716 and a simple chip swap.

Figure 4 gives complete construction details for the adapter. It is very easy to build, and should take all of half an hour. Be sure to use an extra machined-contact DIP socket or socket strip when soldering anything. This will keep the pins properly spaced and aligned should the plastic soften. Plug this extra socket into the cool ends of the pins soldered so it straddles any pins to be heated.

If you use the optional epoxy or super glue, be careful not to get any of it inside the slide switch. Carefully test the switch after the glue sets.

You might later want to add a 28-pin ZIF (zero-insertion-force) socket atop the regular socket. Depending on the pins on the ZIF, this may plug in or have to be soldered.

Figure 5 shows you how to use the adapter on the MPC "ap-ep" card. A very slight modification to the card itself is needed. You have to add a small wire hook to the top of pin one of integrated circuit U10. The easiest way to do this is to tin the top of the pin with a very small amount of solder. Then bend a hook in the end of a resistor lead. Reflow solder the resistor lead in place. Finally, cut away all of the resistor except enough lead to form a small loop.

If you are using a different card or a stand-alone burner, make sure you find a place that is at ground during writing and at +5 volts during reading. Add a small wire hook to this point so the grabber can access this signal. Also be sure that pin 1 will be at +5 volts during read and at +21 or +24 volts during programming. One detail. Note that the 2732 requires a VPP signal of ground for a read and +21 or +24 volts for a write. The 2764 instead requires a VPP signal of +5 volts for a read and +21 or +24 volts for a write. This
read supply difference is the purpose of the diode between adaptor pins 1 and 28. If you are using an oddball EPROM burner, make sure that no "hard" ground shorts this diode or a supply line.

To install the adaptor, put it in the existing 24-pin ZIF socket so that the switch points away from the ZIF handle. You might have to jiggle the handle slightly, center the adapter, and then slide the locking handle home, as there is a very slight "negative clearance" between the open handle and the adapter. Then glomp the grabber onto the wire hook.

Be certain that the EPROM burner or card is configured for a 2732 burn! Never try connecting or disconnecting the grabber with a 2764 in place! As a special note to MPC "ap-ep" users, also be sure both S2 and S3 are in their OFF position, pointing towards the bottom of the card.

If your EPROM burner uses personality modules, be sure to use the 2732 module and not a 2732A module.

Due to lack of space, we're unable to present the entire article at this time. Next month, we'll finish with programming on an older burner, tell you how to capture, modify and test the monitor, and give detailed instructions on how to install the "old" monitor in a DIP.

Fig. 4. Detailed in these drawings and accompanying instructions is the step-by-step procedure you should use to build an adapter for programming a 2764 EPROM in an older EPROM burner set up for burning only 2732's.

### Parts Table

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28-pin machined-contact DIP socket</td>
</tr>
<tr>
<td>2</td>
<td>24-pin machined-contact DIP socket</td>
</tr>
<tr>
<td>3</td>
<td>11-pin machined-contact DIP strip</td>
</tr>
<tr>
<td>4</td>
<td>7-pin machined-contact DIP strip</td>
</tr>
<tr>
<td>5</td>
<td>3-pin machined-contact DIP strip</td>
</tr>
<tr>
<td>6</td>
<td>bare machined-contact DIP pins</td>
</tr>
<tr>
<td>7</td>
<td>extra DIP socket for heat sink</td>
</tr>
<tr>
<td>8</td>
<td>miniature spdt slide switch</td>
</tr>
<tr>
<td>9</td>
<td>1N4001 silicon power diode</td>
</tr>
<tr>
<td>10</td>
<td>mini-grabber test clip</td>
</tr>
<tr>
<td>11</td>
<td>1/2-watt resistor, any value (see text)</td>
</tr>
<tr>
<td>12</td>
<td>4' white #22 stranded wire</td>
</tr>
<tr>
<td>13</td>
<td>2' bare #24 solid wire</td>
</tr>
<tr>
<td>14</td>
<td>2' red #24 solid wire</td>
</tr>
<tr>
<td>15</td>
<td>2' green #24 solid wire</td>
</tr>
</tbody>
</table>

Optional—super glue, epoxy, or silicone adhesive.
Upgrading Apple IIe's ROM Monitor

Part II

Here are fixes to change this model's restrictive monitor to an absolute old reset one, giving users an opportunity to unleash the computer's real power.

By Don Lancaster

Last month in Part I of this article, we discussed Apple IIe firmware, cautioned you about using non-standard ROMs, and began to tell you how to program a 2764 on an old ROM burner. In this concluding part, we continue the burn instructions, tell you how to capture and modify the IIe monitor, and give you a checkout routine for testing your new custom ROM monitor.

Here are the rules on the slide switch use:

- The slide switch points to ground (pin 14) for a low address A12.
- It points to +5 volts (pin 28) for a high address A12.
- Use the LOW switch position for a C or E monitor burn.
- Use the HIGH switch position for a D or F monitor burn.
- You might like to mark a 0 on the left switch position and a 1 on the right switch position.

Capturing the IIe Monitor

Catching the D, E, and F segments of the monitor ROM is rather trivial. In fact, some burner-card software will let you simply move their buffer directly to these locations. But the $C000 to $CFFF monitor capture is a bit tricky and non-obvious. So, Program 1 (not shown here due to lack of space, but available free of charge, along with the source code for KREBF SPELL patch by sending a self-addressed legal-size envelope to MODERN ELECTRONICS, Dept. PROG1, 76 N. Broadway, Hicksville, NY 11801) shows you an Applesloth program called SNATCHMON. What SNATCHMON does is grab four 4K segments of the monitor. It places these on disk under IIEMON.C, IIEMON.D, IIEMON.E, and IIEMON.F names.

You then modify these four segments as needed for your custom monitor. These are saved to disk to be BLOADed to a buffer at $8000 to $8FFF. You can relocate these as needed if your EPROM burner card needs a different buffer area. If you have a stand-alone burner, you can use a modem program to send these binary files out as serial text files, and then use them as needed.

SNATCHMON is surprisingly fast, since all of the actual moves are done using the monitor "M" command. Suitable POKEs activate each move as needed. It is absolutely essential that the Y register be zeroed before a monitor move. To do this, any SNATCHMON move CALL first goes to a five-byte machine program at $7FFB that clears the Y register and then jumps to the actual move routine at $FE2C.

Locations $3C (low) and $3D (high) hold the move starting source address. Locations $3E (low) and $3F (high) hold the move ending address. Finally, locations $42 (low) and $43 (high) hold the starting destination address. After these six locations are POKEd with the correct values, the move can be made.

If you're having trouble converting Applesloth decimal into machine hex, check into the Hexadecimal Chronicles (SAMS #21802) for instant conversions.

To capture the $C100 to $CFFF monitor as IIEMON.C, you have to flip two pairs of soft switches just right. The magic switches involve $C006, $C007, $C00A, and $C00B. Soft switch $C007 turns on the monitor for the $CX00 access, while $C006 turns on the normal I/O. Soft switch $C00B turns on the monitor for $C300 access, while $C00A turns on the normal slot 3 usage. (Note that these switch flips are backwards from page 214 of the Apple IIe Reference Manual. Be sure you correct your reference manual. Very ungodly things happen if you get these actions mixed up. Note that you must write to these locations to activate them, such as with a "C006:00 [cr]" or a
“POKE 49158,0 [cr]”. Reads just won’t hack it.)

There are a few other soft switches used to control the monitor ROM area, but we need not worry too much about these here. Normally, a cold boot is done before you try running SNATCHMON, which flips the switches into their “normal” positions. You then flip soft switches $C007 and $C008 long enough to move the $C100 to $CFFF monitor image to $8100 to $8FFF, and then flip these switches back to their usual $C006 and $C00A positions.

SNATCHMON assumes you have a single disk drive. A “press the spacebar” prompt lets you change diskettes if you want the images of IIEMON.C, IIEMON.D, IIEMON.E, and IIEMON.F to end up on a different diskette.

Modifying the IIe Monitor

The IIe monitor is crammed full and has very little in the way of free bytes. There are a few bizarre bugs, such as a programmer’s ASCII first name that can get executed as op-codes by certain programs accessing < ctrl > - Y at the old location. The percentage of “compatible” II+ programs is far lower than Apple cares to admit, since just about any decent II+ program makes use of “illegal” monitor access. As a general rule, schlock II+ programs are IIe compatible, while creative and useful ones are not.

Areas available for rewriting are the cassette routines, the screen message, and the above-mentioned “Bryan” code. I urge you not to clobber the cassette routines, since cassettes can often salvage a disaster, even after all else has failed.

About the only monitor area that nobody wants to keep is the hole blasting from $C249 to C260. So, as an example of patching the monitor, we will overwrite this hole blaster with code that accesses the old monitor.

The source code appears as Program 2 (see above for availability). It is called KREBF SPELL.SOURCE, and assembles into a KREBF SPELL patch. This module is simple enough that we can forego a flowchart. We’ve used the “new” EDASM format for the source code here.

You can get to location $C249 on a [ctrl] < open-apple dingbat > [reset]. We first blast the warm start location at $03F4 by incrementing it. This tells the Apple that we are to do a cold reboot, rather than letting the applications program in use “capture” the reset back to itself.

Then we run through a timing loop 40 times. Each time through, we stall for a tenth of a second via the monitor WAIT routine at $FCA8. We then check to see if the open-apple key is released. If it is released, we continue with a stock cold reboot, sans the hole blasting. If the key is down, we keep on stalling.

If we timeout to a full four seconds, we instead jump directly to the “old” reset code which, thankfully, remains in the monitor at $FF59. The four seconds was selected to be long enough that a child or an unknowing user is unlikely to accidently trip the old monitor access.

There are some undocumented Apple IIe diagnostics stashed at $C400 to $CFFF that will fail the CD ROM if a checksum of all the bits in the entire ROM does not agree with a magic value.

To pass these diagnostics, the checksum of the KREBF SPELL must be the same as that of the hole blaster bytes it replaced. The magic checksum for these $18 bytes of code is $0BA8. The byte at $825E corrects this checksum to let your altered code pass the diagnostics. Note that any
How to Install Your “Old” Monitor EPROM in an Apple Ile

1. Using the program SNATCHMON, place copies of Ile ROM images onto a work diskette under the filenames IHEMON.C, IHEMON.D, IHEMON.E, and IHEMON.F.

2. Place a copy of KREBF SPELL onto the same work disk. Do this by copying the companion diskette, assembling from KREBF SPELL SOURCE, or by simple hand loading.

3. Cast the KREBF spell this way:
   | BLOAD IHEMON.C <cr>
   | BLOAD KREBF SPELL <cr>
   | BSAVE KREBFMON.C, AS8000, LS1000 <cr>

4. If needed, plug the 2764 adapter into your EROM burner. Be sure the notch points to the handle and that the grabber contacts the correct point in the burner.

5. Erase a 28-pin, 250-nanosecond Intel or Hitachi 2764 EROM using a suitable ultraviolet source.

6. Insert the 2764 EROM into the adapter socket. Be extra careful not to bend any pins. New 2764s will need all their pins “rocked” inwards to fit.

7. Flip the adapter switch to “0” (towards pin 14). Then arm the EROM burner and boot the support code.

8. Verify the erasure. Then load, burn and verify KREBFMON.C as if it were intended for a 2732 EROM.

9. Flip the adapter switch to “1” (towards pin 28). Then load, burn and verify IHEMON.D as if it were intended for a 2732 EROM. Note that the KREBF spell does not get cast on the “D” side.

10. Turn the Apple supply power off and re-hand-load the patch over the monitor code. Details on this appear in the “How To Install Your ‘Old’ Monitor EPROM In An Apple Ile” box.

   Basically, what you do is BLOAD IHEMON.C. Then you BLOAD KREBF SPELL. This patch overwrites the hole blaster with the “old” monitor access timer. Then you BSAVE KREBFMON.C, AS8000, LS1000. All of which casts the Krebf spell on your monitor image.

   You then program your 2764, using KREBFMON.C for the low half (switch left) and IHEMON.D for the upper half (switch right). For this particular use, you need only replace the CD ROM with a 2764 EROM. The EF ROM does not get changed.

   Apple sloth program SNATCHMON, source code KREBF SPELL SOURCE, and machine-language binary patch KREBF SPELL are available ready-to-run on a companion diskette. Also included are modules for the Ile and “new” Ile versions. See the box at the end of this article for full details.

   Since I cannot legally sell you copies of the Ile monitor image if you do not personally own an Ile on which those images are to be uniquely used, these monitor images are not available from me. You will, instead, have to grab your own with SNATCHMON. The painless and automatic process takes all of half a minute.

11. Removing the newly programmed 2764 from the burner and insert it in the newly emptied socket on your Ile, being very careful that the notch goes towards the keyboard and no pins are bent or tucked under. Keep your wrist on the power supply as you do this.

12. Apply supply power and run the following checks:

   (a) Normal cold boot of non-valuable diskette in drive 1 when power is applied.

   (b) Normal reset to Applesloth or applications program on <ctrl > < reset >.

   (c) Normal cold reboot on <ctrl > < open-apple > < reset >.

   (d) Drop into the monitor on <ctrl > < open-apple > < reset > with the < open-apple > key held down for four seconds after release of the reset key. Wait for the beep and the ** symbol.

   (e) From the monitor, do a C006:00 < cr >, followed by a 6 < ctrl > -P. You should get a standard cold reboot.

   BE SURE TO USE THIS “C006:00” < cr > AT THE START OF EACH MONITOR ACCESS OTHERWISE, THE I/O SPACE WILL NOT GET PROPERLY ACTIVATED.

   (f) Do a < ctrl > < closed-apple > < reset >. After some screen flashing, you should get the “KERNEL OK” message.

   (g) Put some opaque tape over the lid of your new “old” monitor. Add a bright dot to remind yourself of this change. Save the CD ROM should warranty repairs ever be needed. This completes installation.

Testing and Extending

Step 12 of the “How To Install” box gives a detailed checkout routine. Be very careful when you swap EPROM for ROM that you bend no pins and that the code dot and notch go towards the keyboard, or front, of your Ile. Save the old CD ROM in protective foam in case warranty repairs are ever needed.
Be sure to turn power off, and remove both ends of the Apple Ile line cord. Always lean on the top of the power supply with your wrist before inserting or removing any integrated circuit. Just to be darn sure, hold the Apple end of the line card in your hand as you do this.

It might be best to program your EPROMs on one 11 + or 1le and test them on a second 1le. This separates programming bugs from strictly operational problems.

Your checkout procedure should include a cold boot, a cold reboot, a cold reboot into the monitor, a cold reboot from the monitor, and a diagnostics pass using the closed-apple key on the le's keyboard.

One very important “gocha”: When you drop into the “old” monitor, you do so with the I/O space disabled and the CD ROM enabled. Always do a C006:00 <cr> as your very first "old" monitor instruction!

This sounds slightly flaky, but I haven't found a really good way around it. Maybe you can. At any rate, this is no worse than good old CALL -151, in fact, it is one keystroke shorter. Besides, it always works—any time, any place, any program, any reason.

If you do not activate the I/O space, all of the usual monitor functions will work in the usual way. The only little problems are that you will have no DOS, printer, or other I/O slot access available. So, be sure to get in the habit of wrapping SC006 on each monitor entry.

There are lots of possible things you can do with a custom monitor. For instance, you can rearrange “Applesloth” to suit yourself. You can also do much of what a “snap-shot” card can for a tiny fraction of the price. To do this, grab the NMI vector and dump everything in sight to the stack, including all registers, all soft switches, hard-to-read locations such as S0200, and some identifying markers. Then exit to some code that uses none of the usual page-zero or screen-scrolling locations.

Let me know what other monitor mods you would like to see. Use the address or phone number in the box. I'll send a free book and product list to anyone who asks.