PLUGGING IN THE EPROM

When you have the switches in the correct position for the the type of EPROM you will be using, you may plug the EPROM into the ZIF (Zero Insertion Force) socket. This is the large blue socket to the left of the guide. The lever above (and below) the socket will securely clamp the leads of the EPROM when it is moved to the right. First, make sure that the lever is to the left. This will allow you to easily insert the EPROM into the socket. If it is a new EPROM, the leads may be bent out a bit, and may have to be bent in slightly. ANOTHER IMPORTANT NOTE: If the EPROM is inserted into the socket incorrectly, you may cause (you guessed it!) catastophic failure of the EPROM. Pin #1 of the EPROM is on the same edge as a notch in the middle of one edge, as shown in figure one. Pin #1 must be toward the top of the ZIF socket. There is a note at the top of the card reminding you of this, and two dots on the left side of the socket showing the location of pin #1. Note that if you have a 28 pin device (2764, 27128, 27256, or 27512), pin #1 is on the upper left side of the ZIF socket. If you have a 24 pin device (2708, 2716, or 2732), pin #1 is the third pin down from the top on the left of the ZIF socket. Thus, if you have a 24 pin device in the socket, there will be two holes visible above the EPROM on each side of the ZIF socket.

INSTALLATION

With power off, plug the PROmGRAMER into slots 3, 4, 5, or 7 of your APPLE][,][+, or //e. Turn on the power, boot the disk, and the "HELLO" program on the disk will give you one page of information (duplicated below). NOTE: If you have not yet done so, make a copy of the disk, and put the original in a safe place. You may use any standard copy program.

PROGRAMMING THE EPROM

There are four programs on the disk labled EPB3, EPB4, EPB5, and EPB7. The number corresponds to the slot that the PROmGRAMER is plugged into. BRUN the appropiate program. For example, if your PROmGRAMER is plugged into slot 4, type "BRUN EPB4". You will then be asked by the computer to type in the type of EPROM you are using. (Type in a number between 0 and 7). You will then be presented with a menu of choices. First, we will mention a few choices that do not show on the screen on this first version, but are available to you.

- K BlanK check. Checks that the EPROM is erased.
- H Help. Goes back to the start of the program.
- X EXits the program and goes to APPLESOFT (type "CALL 2048" to restart).
- M Goes to Monitor. Restart program with "800G"

Choices that show on the screen are:

- L Load memory from the EPROM.
- B Burn the EPROM from memory. (I suppose that we should mention that BURN is colloquial for program.)
- D Display memory
- C Check that programming was correct (you can also use V for Verify.)

The commands K, H, M, and X do not need any parameters. The commands L, B, D, and C (or V) need some further information (at least the first time). Anthropomorphically speaking, you have to tell the computer which area of memory you are using (called the WORKING ARRAY). You do this by specifing the starting address (symbolized by SSSS), and the ending address (EEEE). You also have to tell the computer where you want to start programming the EPROM. You do this by giving is a relative address (PPPP) relative to the start of the EPROM, where the start is 0000. All these addresses are in hexadecimal.

Programming the EPROM basically consists of taking an area of memory, and transferring it to the EPROM. The PROmGRAMER takes care of the mechanics, and it is up to you to make sure to get the correct data into the EPROM. We will show this by example, but first, a word about the driver program (the one that makes the PROmGRAMER work).

The program loads into memory at $800. (The dollar sign signifies that the number following is in hexadecimal). The length of the program is $637, bringing you to $e37. The rest of memory (up to $9600 where DOS begins) is available to you as the working array. A memory map is shown in figure 2.

To program an EPROM, you should have a binary file ready on disk with the necessary information. The length of the data you can fit in depends on the specific EPROM you are using. Following is a table of EPROM types, the length of the file that can fit in, and programming times;

EPROM TYPE	LENGTH (in hex)	LENGTH (in dec.)	TIME TO PROGRAM (min. and sec.)
2708	400	1024	0:51
2716	800	2048	1:42
2732	1000	4096	3:24
2764	2000	8192	6:50
27128	4000	16384	13:40
27256	8000	32768	27:18
27512	10000	65536	54:36

For our first example, let's try programming a 27128 with a unique pattern that tests every possible combination. Looking at the above table, we can see that the maximum length of the file is $4000 bytes. On the disk that we sent you is a program called "TEST PATTERN". This just consists of a sequence of numbers from 0 to $FF, repeating as necessary to fill the $4000 byte range. First, we set the switches, plug in the EPROM, and plug it into a slot (REMEMBER - POWER OFF!). Turn on the power, boot DOS, and type;

```
BLOAD TEST PATTERN
```

This pattern will load starting at location $1000. Now, we need the PROmGRAMER driver program, so (assuming the card is in slot #4) we type;

```
BRUN EPB4
```

Since we are using a 27128, type the number 4, which is the type number for a 27128.

We will now "burn" the EPROM with the memory range from $1000 to $4FFF, starting at EPROM location 0, so we type:

```
B 1000 4FFF 0000
```

Note that it is necessary to put in the blank spaces, and four digits per number.

Now, make yourself a cup of tea, and find something to pass the time. Each byte takes 50 milliseconds to program (0.05 seconds), so we have a wait of about .05 seconds times 16384 = 819.2 seconds, or 13 minutes 39.2 seconds. Just so you can tell that something is happening, the computer will beep at you every time is completes a page of memory (256 bytes), about every 13 seconds.

After programming is complete, the PROmGRAMER will check if the job was done correctly. It will compare each byte of memory with the corresponding location in the EPROM, and verify that both are the same. If not, the computer will beep and type the message "ERRXXXX", where XXXX is the address of the offending byte. If there are no errors, the computer will beep after each page of memory is verified. When the cursor reappears, the job is done.

To protect ourselves from angry calls, we have to tell you that we recommend that you power down before removing the EPROM. As a practical matter, we have never blown anything by removing the EPROM with power on, but if you DO blow something, forget where you read this. However, you MUST turn off power before removing the card, or dire consequences will follow.

If you remove the EPROM without shutting of power, and you want to program another with the same information, all you need to type is "B". Since the parameters have not changed, you do not need to retype them. If you forget to type in the parameters the first time, you will get a beep and an "ERR" message.

READING EPROMS

Turn off the computer to assure yourself that the test pattern in memory has disappeared. Turn the computer back on, and BRUN EPB4. Type 4 (the eprom type), and;

```
L 1000 4FFF 0000
```

This command will load memory locations $1000 through $4FFF with the information in the EPROM starting at EPROM location zero. To display the memory, type D. (no parameters are necessary, as they are the same as the last entry). You should see the test pattern appear. (You may stop the listing with a control-S, and continue it with another keypress). The information you are looking at was stored inside the EPROM. At this point, you can be sure that the PROmGRAMER and your technique are working.

PROGRAMMING THE 27256

For some reason, most people who get into programming EPROMs start their files at $2000. That is a nice round number, it won't interfere with the PROmGRAMER driving program,

as it does seem to be a standard, so why not continue to use it? Well, we'll tell you why. If you start at $2000, the file for a 27256 will end at $9FFF. We all remember (don't we?) that DOS resides starting at $9600, so we have a conflict. In this type of conflict, no one wins. We must avoid overwriting DOS. If we start the file at $1000, it will end at $90FF neatly avoiding problems, so let's set a standard: All files will start at $1000 (naturally, you may start at another location if it is necessary for a particular job).

One other thing; under DOS 3.3 the maximum file length you can save to disk is $7FFF bytes, one byte less than that needed for a 27256 (amazing how often things like that seem to happen). We can do one of two things, save one byte less than we need, or change DOS. If you type POKE -22172,255, DOS will now accept files up to 64 Kbytes long. We suggest that you do not INIT any disks after making this change, because your DOS is no longer standard (until you re-boot).

PROGRAMMING THE 27512

The 27512 can hold 64K of memory, which is the entire memory of the standard APPLE][. Obviously, most of that is something we would not want to program, so we must do it in two steps. Assuming the files are saved as "FIRST HALF" and "SECOND HALF", we would proceed somewhat as follows:

```
BLOAD FIRST HALF, A$1000
BRUN EPB4
7  [EPROM TYPE]
B 1000 90FF 0000
   [after 28 minutes]
X [goes to APPLESOFT]
BLOAD SECOND HALF, A$1000
CALL 2048 [since the program is still intact]
7
B 1000 90FF 8000
```

YOU'RE ON YOUR OWN, NOW

Well, not quite, we are available to give you any necessary assistance. Call us at (805) 685-1931. We may be moving after December 15, so you might get operator intercept which will give you the new number.