

---

# Chapter 12

---

## ROM and Static RAM

In order to do some useful work, the SK68K needs some RAM and ROM. The ROM contains a debugging program called HUMBUG and a simple Basic interpreter; the RAM is needed as a temporary memory to hold various data needed by HUMBUG.

### 12-1. Discussion

A fully configured SK68K system contains both static RAM (also sometimes called SRAM) and dynamic RAM (called DRAM). Static RAM circuitry is generally very simple and inexpensive, but only for small amounts of memory. Whenever a very large amount of memory is needed, dynamic RAM is the only economical alternative. We will examine dynamic RAM in a later chapter; for now, however, we will concentrate only on static RAM.

Since static RAM is so much simpler - needing just two ICs - installing a small amount of it at this time gives us a fast way of getting the SK68K operating. For that reason, HUMBUG and the ROM-based Basic are specially configured to use only the static RAM. We will add DRAM later, after the SK68K is at least partly operational; in that way, we will be able to use the remaining parts of the system to debug the DRAM circuitry if there are any problems. We will not actually need the DRAM until we are ready to run the disk system; running large programs of the type best suited for DRAM is really not practical until we have some means of saving and loading them on a disk.

Fig. 12-1 shows the circuitry for the EPROM and the static RAM. The 68000's data bus is 16 bits wide, but no one makes EPROMs and RAMs which have 16 data lines. The solution is to use two 8-bit-wide EPROMs and two 8-bit-wide static RAMs, each of which holds 8 of the 16 bits.

The two EPROMs are shown at the top, and the two static RAMs are shown at the bottom. Three kinds of EPROMs can be used, though both

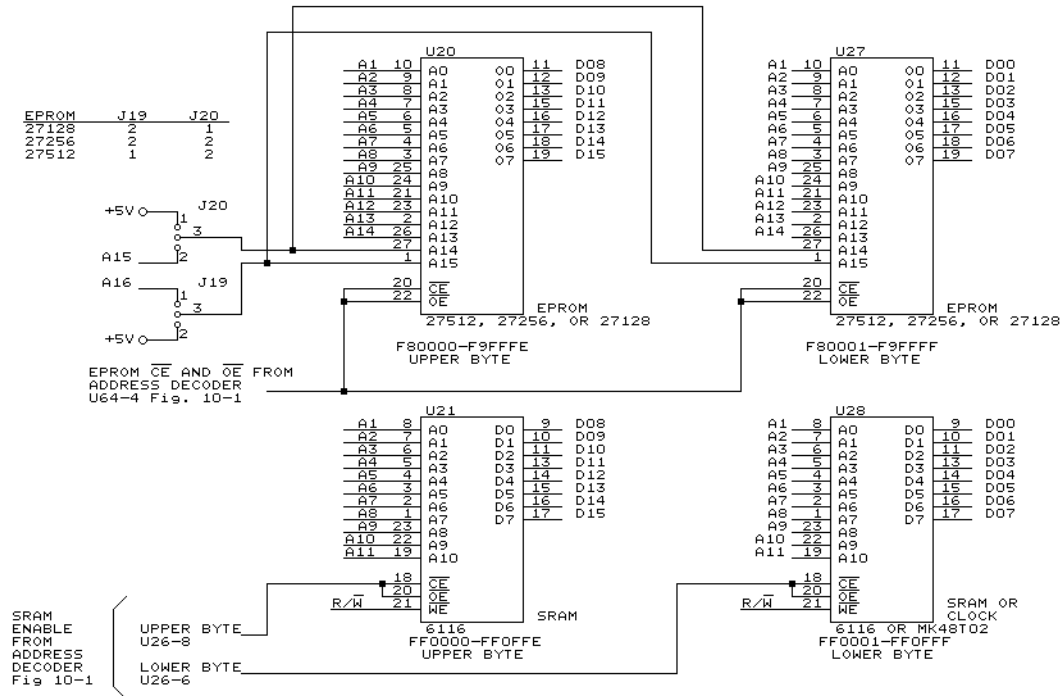


Fig. 12-1. EPROM and static RAM wiring.

EPROMs of a pair must be identical. A pair of 27128s holds a total of 32K bytes; a pair of 27256s holds 64K bytes; a pair of 27512s holds 128K bytes. Any one of these would be big enough to hold HUMBUG and Basic, but the prices on memory ICs have been so unstable that your SK68K may contain any one of these - it depends on which is cheapest at the moment. Jumpers J19 and J20 simply determine which kind of EPROM can be used, because larger EPROMs require more address lines.

The two static RAMs are shown at the bottom. The simplest version of the SK68K uses a pair of 6116s, which provide a total of 4K bytes of memory. But it is possible to replace one - or both - with a Mostek MK48T02, which also contains a static RAM and has the same pinout as a 6116, but has two additional features - it contains a built-in clock/calendar (which replaces the top 8 locations of the RAM), and it has a built-in lithium battery which powers both the clock and the RAM when the computer is turned off. The battery is rated for at least 31,000 hours of operation, or slightly more than 3-1/2 years. In actual use, it would probably last longer since the battery only powers the clock when the computer is off. (Note that the MK48T02, if used, must be inserted into U28, since the clock software in SK\*DOS expects to find it in the lower byte.)

As to the wiring, the two left ICs (left in Fig. 12-1; physically, they are really located toward the back of the actual board) connect to D8 through D15; the two right ICs connect to D0 through D7. Thus the two left ICs handle what is called the *upper byte*, whereas the two right ICs hold the *lower byte*. This is sometimes confusing and needs some explanation. When the 68000 stores a word (two bytes) from an internal register into memory, the

left byte (the more significant byte, also called the upper byte) goes into memory first, and is always in an even-numbered location. The right byte (least significant or lower byte) goes into memory next, and goes into the next higher odd location. For example, the number \$1234 might be stored as \$12 in location \$3500 and \$34 in location \$3501. This is the more logical way to do things, but it is confusing for two reasons: (1) the upper byte is actually stored in the lower address, and (2) this is the opposite of what Intel processors do.

Both EPROMs are controlled by the same  $\overline{CE}$  (chip enable) and  $\overline{OE}$  (output enable) signals, whereas there are separate enable signals for the static RAMs. Reading both EPROMs at the same time does no harm - if the 68000 only wants one byte, it simply ignores the other half of the data bus. But writing to the static RAM requires two control lines to make sure that writing to one RAM does not store garbage in the other. Note also the difference between  $\overline{CE}$  and  $\overline{OE}$  - when a read is done from RAM or EPROM,  $\overline{CE}$  enables the IC and starts the read process, but no output gets to the bus until  $\overline{OE}$  is asserted. In many systems,  $\overline{CE}$  is used to put the entire chip into a low power mode when not being used; in the SK68K we simply control both together so it always switches into low power mode after every access.

Note that on all four memory ICs, the IC's A0 connects to A1 on the address bus, A1 connects to A2, and so on. Part of the reason is simply that A0 does not exist on the address bus; part comes from the way memory is addressed on the SK68K. Consider the static RAM, for example, which is addressed starting at address \$FF0000. The bytes of this RAM are stored as follows:

```
$FF0000 is in U21 location 0
$FF0001 is in U28 location 0
$FF0002 is in U21 location 1
$FF0003 is in U28 location 1
$FF0004 is in U21 location 2
$FF0005 is in U28 location 2
```

and so on, with all even addresses (the 'upper byte') in U21, and the odd addresses ('lower byte') in U28. The address lines are 'shifted over one bit' because shifting a binary number to the right by one bit divides it by two. For example, location \$FF0008 ends with the bits 1000; it would be in location 100 of U21. Location \$FF0009, on the other hand, ends with the bits 1001; it would also be in location 100, but in U28. Thus the last bit of an address (0 in \$FF0008, 1 in \$FF0009) tells us which IC it is stored in, while the preceding bits specify the location in that IC.

## 12-2. Construction

Now that we understand how the EPROM and static RAM circuitry works, let us actually connect it. Remove the Molex pins from the U21 and U27 sockets, and install the following components:

U26	74LS32 and its socket
C12 and C66	0.1 $\mu$ F disc capacitors

U20	EPROM marked 'upper' and its socket
U27	EPROM marked 'lower'
U21	6116 2Kx8 static RAM
U28	6116 2Kx8 static RAM and its socket
J19 and J20	three-pin header strips

Examine the two EPROMS and position the J19 and J20 jumpers to match the type of EPROM used, following this table:

EPROM	J19	J20
27128	2	1
27256	2	2
27512	1	2

Place jumper J25 in position 2 to address the EPROM starting at location 0.

## 12-3. Testing

Now turn on the power.

The important sign that all is well is that the HALT LED goes off after about a second. If not, then go back a step, recheck that all worked before, and then recheck all connections and parts installed since then.

The design of 68000 computers is such that they tend to halt if something goes wrong. If the HALT LED is off, that is a pretty good sign that nothing major is wrong, even though the computer is still not fully operational.

Use the LED probe to check a few signals. First of all, all data bus lines should have pulses (indicated by the LED dimming when you connect to the line). Likewise, all address lines from A1 through A18 should show lots of pulses, whereas A19 through A23 should not dim the LED at all. If the HUMBUG program in the EPROM is running, it is accessing mostly itself (EPROM locations \$F80000 and above) and static RAM (locations \$FF0000 and above). Since both \$F8 and \$FF start with the five bits 11111, we would expect the five highest bits of the address bus - A23 through A19 - to be constantly high, even though the others may change.

You can confirm this by testing the outputs of the address decoder. You will note a lot of pulses, indicated by the LED dimming, on U63 pin 19, which decodes all addresses above \$F80000. U64 will have pulses on pin 4 (which selects the EPROM), pin 7 (which drives U64b), and pin 10 (which selects the static RAM.)

What you don't see on the LED (though a good scope will reveal them) are tiny low pulses on A23 through A19, as the 68000 is trying to access some error vectors because it senses a bus error - no DUART is installed. Likewise, you can see some tiny pulses on U64 pin 5, as HUMBUG is desperately trying to access the expansion slots in the hope that there is a video board there to report the error on.