

---

# Chapter 18

---

## DRAM Circuitry

In this chapter we describe and build the actual DRAM circuits in our SK68K computer. Since we have already discussed the general principles behind DRAM circuits, we will discuss the specific DRAM circuitry of the SK68K.

### 18-1. Discussion

#### Block Diagram Description

The block diagram of the complete DRAM circuitry is shown in Fig. 18-1. The overall timing of refreshing is handled by the DRAM control circuits, which receive RSHAS and the DRAM enable signal from the address decoder, the AS address strobe, and a clock, and which generate the RAS, CAS, and DTACK signals, as well as three enable signals which control three sets of tri-state buffers.

The three sets of tri-state buffers are separately enabled by the DRAM control circuits, with only one set of buffers enabled at any one time. In this way, three different inputs can be combined onto one set of pins. When the CPU is accessing memory, these buffers would work in this order:

1. The 68000 outputs an address, the address decoder recognizes a DRAM address and sends the DRAM enable signal to the DRAM control circuits.
2. The control circuits enable tri-state buffers A (with B and C disabled) to send nine bits of the address to the DRAM ICs, and then pulse RAS to latch them in the row address buffers within each DRAM.
3. The control circuits then enable tri-state buffers B (with A and C disabled) to send the other nine bits of the address to the DRAM ICs, and pulse CAS to latch them in the column address buffers within the DRAM.

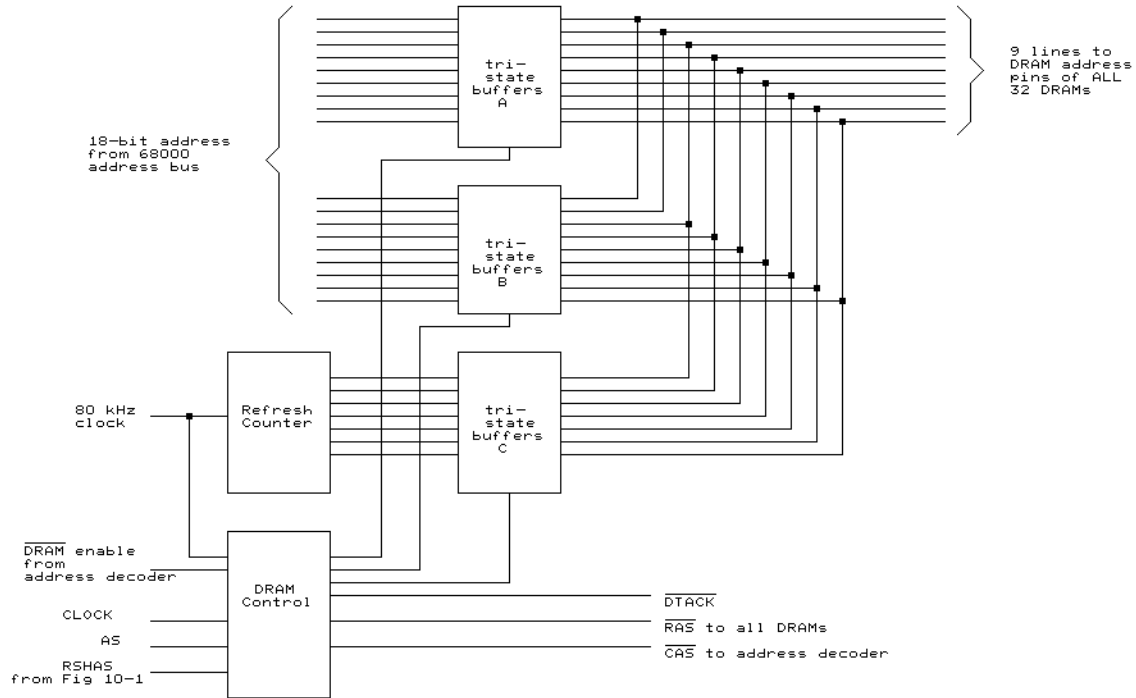


Fig. 18-1. DRAM section block diagram.

The  $\overline{RAS}$  signal goes to each DRAM IC so that all ICs, even those not being used, receive a row address. But the  $\overline{CAS}$  signal goes back to the address decoder, in Fig. 10-1, which steers the  $\overline{CAS}$  to the appropriate group of ICs.

The completed SK68K computer uses 32 256Kx1 DRAM ICs to provide a total of one megabyte of RAM. These ICs are organized in four banks of 256K bytes each as follows:

- U38 through U45 hold all the odd bytes for the first 512K
- U53 through U60 hold all the even bytes for the first 512K
- U67 through U74 hold all the odd bytes for the second 512K
- U80 through U87 hold all the even bytes for the second 512K

When a particular byte (or 16-bit word, in the case of a two-byte transfer) is accessed, only one or two of these banks need be enabled. Thus the address decoder does the final selection, based on the state of A19,  $\overline{UDS}$ , and  $\overline{LDS}$ , and sends  $\overline{CAS}$  only to those banks being accessed. The DRAM ICs use  $\overline{CAS}$  as their main chip enable, and only those ICs getting  $\overline{CAS}$  do an actual read or write.

Refreshing is initiated by the 80 kHz clock input to the refresh counter and the DRAM control circuits. Once every 12.5 microseconds, a clock pulse arrives, increments the refresh counter to a new row address, and sends a refresh request to the DRAM control circuits. The control circuits then wait until the 68000 stops using the memory and begin the refresh sequence by

enabling tri-state buffers C (with A and B disabled) to send the refresh address to the DRAM chips, followed by a pulse on the  $\overline{\text{RAS}}$  line. Since  $\overline{\text{RAS}}$  as well as the address lines go to all 32 DRAMs, all the DRAMs are refreshed at the same time.

The complete refresh cycle for all 128 rows takes 1.6 milliseconds (12.5 microseconds between clock pulses times 128 rows), which is somewhat less than the 2 milliseconds specified for most DRAMs, but there is nothing wrong with refreshing the DRAMs more often than necessary.

### DRAM Operation During CPU Accesses

Let's begin our discussion with the timing circuits shown in Fig. 18-2, and let's assume flip-flops U49a and U49b are reset, which is the condition most of the time (for U49a is reset every time address strobe AS goes low). Since U49b is reset, its REFRESH output is low and  $\overline{\text{REFRESH}}$  is high. Furthermore, since U49a is also reset, the CAS and DRAM DTACK outputs are both high.

When the 68000 wants to access DRAM to either write into it or read from it, it places a valid DRAM address on the address bus. The address decoder (Fig. 10-1) recognizes the address as referring to the DRAM, and sends out the DRAM enable signal which goes to Fig. 18-2 and starts a DRAM cycle. (The signal travels a long distance on the board, and C68, a 33 pF capacitor, is used to reduce its noise pickup.)

The active-low DRAM signal goes low at this point; since pin 10 of U51c is also low as we will see in a moment, U51c sends out a low pulse to U37c pin 10.

At this point it pays to recap for a moment. U51c is actually an OR gate, but it is shown as an AND gate with bubbles on both its inputs and outputs. This notation is used because its job in this circuit is not to *or*, but to *and* two signals. In this case, when both of its inputs go low - and only then - its output goes low. U37c, on the other hand, is a NAND gate but it is shown as an OR gate with bubbles on its inputs. This notation is used because its

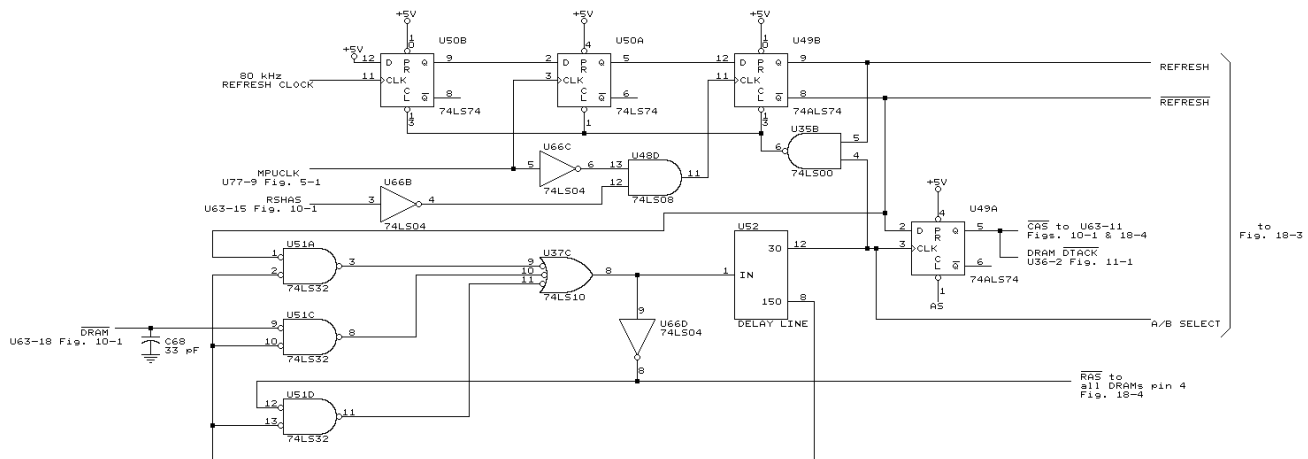


Fig. 18-2. DRAM timing circuits.

job is not to *and* three signals, but to *or* them. Its job is to output a high whenever any one of its inputs goes low. In other words, there are three conditions under which pin 8 of U37c can go high.

As we mentioned two paragraphs ago, when the 68000 wants to access DRAM, it sends out the RAM signal which, in turn, makes U51c pin 8 go low. As a result, U37c pin 8 goes high. This signal now does three things.

First, it is inverted to a low by U66d to generate the RAS signal. This is the *row address strobe* which goes to the DRAM ICs to tell them to accept a row address. The same signal also goes back to U51d pin 12; since pin 13 is already low (just like pin 10), U51d pin 11 also now becomes low. This provides a second low to U37, making sure that it continues to output a high on pin 8 even if the RAM signal should disappear at this point. This is not important right now, though, since the RAM signal is not going to disappear until after the DRAM operation is completed (this circuit is only needed during refreshing). Finally, the high on U37c pin 8 also goes to U52.

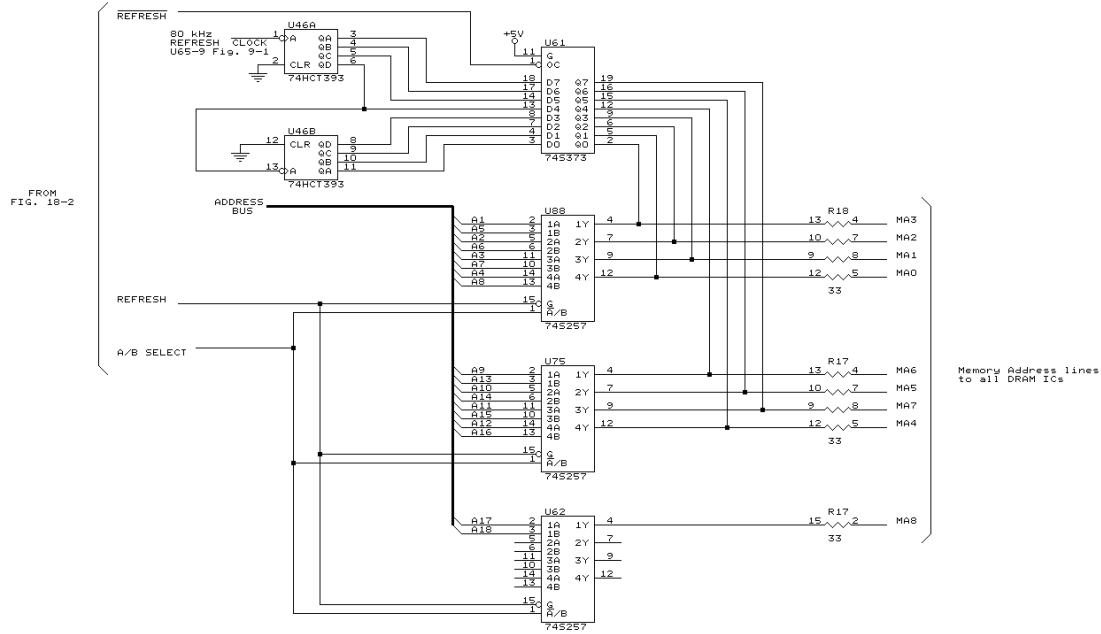
U52 is called a *150-nanosecond delay line*, and it does exactly what the name implies - it delays signals. It has several outputs, of which we only use pins 12 and 8. Whenever a logic signal is applied to its input on pin 1, it is delayed and appears on the outputs a specified delay time later. The total delay line is specified as 150 nanoseconds (ns), which means that the input comes out the last output, pin 8, 150 ns after it entered. But the delay line also has an intermediate output on pin 12, which provides only a 30 ns delay. (When you really think about it, 30 ns is a very short time. For example, a beam of light - the fastest thing we know of - only travels about 30 feet in 30 ns!)

Up until now, however, the input to the delay line was a low, and so both of its outputs have been low. U52 pin 8 therefore has been sending a low back to U51c pin 10 and U51d pin 13, which were needed to get everything started. U52 pin 12, on the other hand, has been sending out a low to flip-flop U49a, to U35B pin 4, and to the A/B SELECT line. After the 30 ns delay, however, U52 pin 12 goes high. This changes the A/B SELECT signal from a low to a high, and also clocks flip-flop U49a. Since pin 2 of the flip-flop is already high (since flip-flop U49B was reset and therefore REFRESH was high), the flip-flop now sets and sends out a low on CAS and DRAM DTACK.

Let's now jump ahead to Fig. 18-3 and see what is happening in the DRAM address multiplexers. As we mentioned last time, the DRAM address pins receive three separate addresses. During normal memory operation, they receive first a row address, followed by a column address; during refreshing, they receive a refresh address. The multiplexers shown in Fig. 18-3 select which of these three addresses is applied when.

Up until now, REFRESH has been low and REFRESH has been high. The circuitry at the top of Fig. 18-3 generates the refresh address, which can be sent to the DRAMs through a set of tri-state buffers in U61 if their OC (output control) input goes low. But since REFRESH is high, this keeps OC high and therefore prevents the refresh address from getting to the DRAMs.

Instead, the low REFRESH signal is applied to the G (gate) chip select inputs of U88, U75, and U62, enabling their circuitry. Each of these three ICs is a "quad two-input multiplexer", meaning that it contains four multiplexers, each having two inputs. The A/B select input on each IC selects



DRAM address multiplexers.

which of the two inputs is sent to the output of each multiplexer. Scanning down the inputs of U88, for example, A1 goes to output MA3 if the A/B select input is low, or A5 goes to MA3 if the A/B select input is high; similarly either A2 or A6 goes to MA2, depending on the A/B select input, and so on. Notice that the A inputs and MA outputs seem to be mixed up in a crazy order, which seems as though the memory is going to be very confused. In actual operation, it simply means that every time the 68000 stores something into memory it will go into what looks like the wrong location; but the next time the 68000 wants to read it back, it will be read back from the same wrong location and so the correct data will come back out.

At the beginning of the operation, A/B SELECT was low, and so each of the multiplexers chose one set of nine bits to send to the MA outputs. But after the delay line outputs a high A/B SELECT, the multiplexers switch and send the other nine address bits to the MA outputs. The first set of nine bits was the row address; the second set is the column address. Although it looks as though the column address comes out just 30 ns after the row address, actually the delay is somewhat greater. The row address is applied to the DRAM ICs as soon as the 68000 starts its memory operation; the column address is not applied until after the address decoder has recognized the DRAM address and sent out the RAM signal, which must then go through U51c and U37c before even entering the delay line.

Up until now, the operation has been as follows:

1. The 68000 sends out an address; since REFRESH and A/B SELECT are both low, nine bits of the address go to the DRAMs as a row address.

2. The address decoder sends out  $\overline{\text{RAM}}$  which starts the DRAM circuitry.
3.  $\overline{\text{RAS}}$  goes low.
4. After a short delay, A/B SELECT goes high and sends the other nine address bits to the DRAMs; this is now the column address.
5.  $\overline{\text{CAS}}$  goes low.
6. DRAM  $\overline{\text{DTACK}}$  goes low.

Note that  $\overline{\text{DTACK}}$  normally signals the 68000 that a data transfer is completed, and yet the DRAM access is nowhere near being finished. In an effort to keep the computer going at maximum speed,  $\overline{\text{DTACK}}$  is sent to the DTACK circuitry (U36, shown in Fig. 11-1) before the DRAM actually finishes; the 68000 doesn't respond for another few clock pulses and so the DRAM will have enough time to finish before the 68000 continues. Note that the timing of the entire DRAM circuitry is very carefully thought out; since most SK68K operations involve the DRAM, squeezing extra speed out of this circuit is very important.

Let's now return to Fig 18-2. After the 150 ns delay, U52 pin 8 goes high, which makes U51c pin 10 and U51d pin 13 both high. As a result, all inputs to U37c go high and so its output goes low. After passing through U66d, this makes  $\overline{\text{RAS}}$  go high again. Meanwhile delay line U52 is now processing the low. After 30 ns, the low on U52 pin 12 makes A/B SELECT return low; it also changes the clock signal to U49a from high to low, but the flip-flop does not react since the 74ALS74 flip-flop only responds to a rising edge of the clock. Thus the flip-flop stays set, and continues outputting  $\overline{\text{CAS}}$  and DRAM  $\overline{\text{DTACK}}$ .

After the delay line completes the 150-ns delay, its pin 8 goes low and the DRAM circuits are ready for another memory access. Meanwhile, the 68000 finishes its memory access also; it then turns off the AS address strobe, with the result that U49A finally resets, and both  $\overline{\text{CAS}}$  and  $\overline{\text{DTACK}}$  go back high or off.

### RAM Operation During Refreshing

Let's now look at how the DRAM circuits operate when refreshing. First, the 80 kHz REFRESH CLOCK (from the bus error circuit of Fig. 9-1) is sent to the refresh counters, U46a and U46b, in Fig. 18-3. As mentioned last time, refreshing a DRAM involves reading 128 rows once every 2 milliseconds. Although only 128 rows are needed, U46a and U46b are each divide-by-16 counters, so together they make a divide-by-256 counter. All eight of their outputs are sent to the DRAM address lines through U61, even though only seven outputs are needed. The eighth output does no harm, however, and is there for possible future expansion.

Since the period of the 80 kHz signal is 1/80000 second, or 12.5 microseconds, the refresh counters complete a total of 128 counts in 128 x 12.5, or 1600 microseconds. This is 1.6 milliseconds, well within the ratings of the DRAM chips which require refreshing at least once every 2 milliseconds.

Once every 12.5 microseconds, when pin 9 of U65b goes high, this signal clocks U50b in Fig. 18-2, which therefore sets (since its D input is connected to +5 volts) to signal the DRAM circuits that it is time to do a refresh. The Q output of U50b therefore goes high, so that the next rising edge of the

MPUCLK signal (which might be anywhere from 8 to 16 MHz, depending on CPU speed) sets U50a. Its Q output now also goes high, sending a high to the D input of U49b. The circuit now waits for two other events: for the RSHAS pulse to go low (which is inverted into a high by U66B) and for MPUCLK to go low again. When this occurs, U49b finally sets, and this event finally starts the refresh cycle. Note that it was not sufficient for the 80 kHz signal to go high - the circuit waited for the correct sequence of MPUCLK and RSHAS before actually starting the refresh. The reason is that a refresh can interfere with the 68000's use of the DRAM memory unless the refresh is timed just right. This circuitry is designed to delay a refresh until the end of an address strobe; the idea is to try to do a quick refresh just after the 68000 has finished accessing memory and so try to squeeze the refresh into an unused time period. (During normal operation, the RSHAS signal - which comes from U63 in Fig. 10-1 - is the same as the AS strobe, so timing is synchronized with the end of AS. But when the 68000 is waiting for a DTACK from a board plugged into one of the expansion connectors, U63 substitutes a steady low to allow the DRAM circuitry to refresh without waiting for a real AS strobe.)

In any case, refreshing begins when U49b finally sets. This switches the REFRESH signal from a low to a high, and switches REFRESH from a high to a low. As shown in Fig. 18-3, when REFRESH goes high it turns off the three multiplexers (U62, U75, and U88); when REFRESH goes low it enables U61, so that the DRAM ICs receive the refresh address instead of the normal column or row address from the address bus.

Back in Fig. 18-2, however, the REFRESH signal is also sent to pin 1 of U51a. This starts exactly the same memory cycle as was started by the RAM signal from the address decoder, so that the delay line receives first a high and then a low, just as in normal operation. But this time there are two differences: first, the REFRESH signal is now low, and so the 30-ns output on pin 12 of delay line U52 cannot set U49a. Thus no CAS or DRAM DTACK is generated during a refresh. This means that only a row address is sent to the DRAMs, no column address. Furthermore, DTACK is not needed since the 68000 is not involved in refreshing, and doesn't even want to know that refreshing is occurring. The second difference is that the 30 ns output of the delay line is also sent to pin 4 of U35b. Since pin 5 is already high (because REFRESH is high), U35b pin 6 goes low, thereby resetting flip-flops U50b, U50a, and U49b. This ends the refresh cycle.

Finally, we need to look at how the 32 DRAM ICs are actually connected; Fig. 18-4 shows the connections to just U38, one of the DRAMs. First, each 256K DRAM IC has nine address lines which connect to MA0 through MA8. As shown in Fig. 18-3, these nine lines each come through a 33-ohm resistor, a rather unusual practice in normal digital circuits but actually quite common in memories. These resistors are used primarily to reduce overshoots and undershoots of voltage (above +5 volts and below 0 volts) which would otherwise exist on these address lines. The problem is basically caused by the fact that the memory address lines each go to 32 ICs. The wiring for these lines is therefore quite long and complex and thus represents a fairly large capacitance to ground. The buffers driving these lines would normally feed rather large current surges into this wiring, which would result in overshoots and undershoots. 33-ohm resistors are also

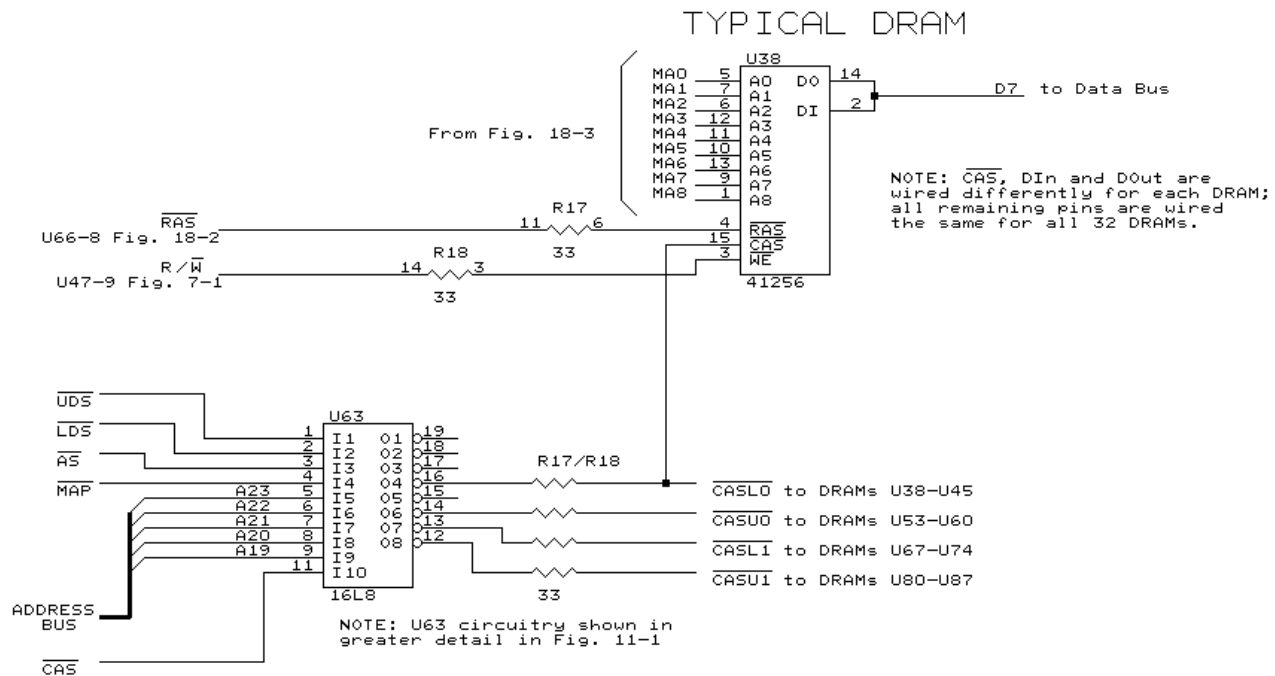


Fig. 18-4. DRAM IC wiring.

found on the  $\overline{R/W}$  and  $\overline{RAS}$  lines, which also go to all 32 DRAMs, as well as the  $\overline{CAS}$  lines. Only the memory data lines do not contain the resistors, since each data line goes to only two ICs.

The  $\overline{RAS}$  signal from Fig 18-2, and the  $\overline{R/W}$  signal (from the 68000) are also applied to each and every DRAM IC in the entire system. That means that all DRAMs accept the same row address, and also all receive the read/write signal at the same time (although they don't actually read or write unless they also receive the  $\overline{CAS}$  signal).

The  $\overline{CAS}$  signal leaving Fig. 18-2 does not go directly to the DRAMs; instead it goes back to the address decoder (originally shown in Fig. 10-1, but also partly shown at the bottom of Fig. 18-4.)  $\overline{CAS}$  is used only during actual memory accesses by the 68000, not during refreshing. At that time, the address decoder must decide which group of DRAMs is actually being addressed. The 32 DRAMs are divided into four groups of eight ICs:

- (a) the even (or high-order) bytes of the first 512K, U53 through U60, which connect to  $\overline{CASU0}$ ,
- (b) the odd (or low-order) bytes of the first 512K, U38 through U45, which connect to  $\overline{CASL0}$ ,
- (c) the even (or high-order) bytes of the second 512K, U80 through U87, which connect to  $\overline{CASU1}$ , and
- (d) the odd (or low-order) bytes of the second 512K, U67 through U74, which connect to  $\overline{CASL1}$ .

U63, the address decoder PAL, decides which group (or groups) of DRAMs to enable, based on the status of upper data strobe UDS, lower data

strobe  $\overline{\text{LDS}}$ , and address lines A19 through A23, and then passes the  $\overline{\text{CAS}}$  signal to the appropriate group or groups. Note that it might send  $\overline{\text{CAS}}$  to two groups of DRAMs at the same time if a 16-bit data transfer is needed from both an even and an odd byte at the same time.

Finally, within each group of eight DRAMs, each IC is connected to a different bit of the data bus so that every data transfer involves either eight or sixteen DRAMs, all writing or reading a different bit of the data bus. Since the DRAMs have separate data in and data out pins, these are connected together as shown in Fig. 18-4. Fig. 18-5 shows a pictorial view of the 32 DRAMs on the printed circuit board, identifying which IC connects to which data line, and which group connects to which  $\overline{\text{CAS}}$  line.

## 18-2. Construction

It is now time to build the DRAM portion of the board. Begin by checking that the following components have been installed in prior steps:

U35, 74LS00 quad NAND gate and its socket # U37, 74LS10 triple 3-input NAND and its socket # U48, a 74LS08 quad AND gate and its socket # U51, a 74LS32 quad OR gate and its socket # U66, 74LS04 hex inverter and its socket # C68, a 33 pF disk capacitor # R12, a 10K resistor # R17 and R18, 33-ohm DIP resistor packs # U65, a 74LS390 dual decade counter and its socket.

Recheck C68 to make sure it is 33 pF, and not 0.1  $\mu\text{F}$  like almost all the other disk capacitors on the board.

Now install the following:

U49	74S74 dual flip-flop and its socket
U50	74LS74 dual flip-flop and its socket
U52	150-ns delay line, soldered directly to the board
U62, U75, U88	74S257 quad 2-input multiplexers and their sockets. U62 uses a special IC socket with a built-in decoupling capacitor, as there was no room on the printed circuit board to place a separate capacitor right next to it.
U46	74HCT393 dual divide-by-16 counter and its socket
U61	74S373 8-input tri-state buffer and its socket. Make sure that this particular IC is not made by TI; if necessary, interchange with U19 to make sure it is made by a different manufacturer.

Although a TI-branded 74S393 works just fine as U19, the DRAM refresh circuitry has some very critical timing, and 74S373 ICs made by TI do not seem to operate well in this circuit - we have had good luck with units made by National Semiconductor and others.

We are now ready to install more components, but much of the wiring in this area is very close and we must take special precautions to guard against accidental short circuits. When installing the following components, install the IC sockets first, and after every group of eight or so,

recheck the wiring and then turn on the computer to make sure it still works (make sure to remove all loose wires or solder before turning on the power). Although this is not a foolproof check, it does help to narrow down the cause of most problems as soon as possible after they happen. If at any point the computer suddenly stops working, recheck all new soldering and components and look for accidental solder joints. Here are the components to be installed next:

U38-U45 and U53- Sixteen 41256 dynamic RAM ICs  
U60

U67-U74 and U80- Another sixteen 41256 DRAMs if the optional second  
U87 512K of memory is to be installed.

Use 150-nanosecond ICs at 8 or 10 MHz, 120-nanosecond ICs at 12.5 or 16 MHz clock speeds. Sixteen ICs will provide 512K of RAM and is the minimum number that can be installed at this time. Even if you install only 16 DRAMs, it is a good idea to install all 32 sockets to avoid having to pull the board out of the cabinet later.

Next, install

C15-C46	thirty-two 0.1 $\mu$ F disk capacitors near pin 1 of each of the 32 DRAM sockets
C47, and C49 through C54	seven 0.1 $\mu$ F disk capacitors along the bottom edge of the board below the DRAM sockets
C56	0.1 $\mu$ F disk capacitor between U75 and U88.

## 18-2a. Additional Construction Step

If you have an early production PC board, your circuit board has an error for which we apologize. As wired on the board, U48d-12 is connected directly to AS, rather than going through U66b to RSHAS. This is a connection which worked well on the original system. But recent production XT-compatible monochrome video boards have a tendency to output a long wait signal to the bus while they complete internal operations. As originally wired, the SK68K would therefore delay DTACK while waiting for the video board to finish its operation. During this time, there would be no AS strobes, and therefore no refreshing.

As described earlier, the modified SK68K circuit now performs a refresh without waiting for AS while waiting for a video board. To implement this change (which may not be necessary except when used with new video boards), perform the following:

Cut the trace leading to U48d pin 12. Then install a jumper from U63 pin 15 to U66b pin 3, and another jumper from U66b pin 4 to U48d pin 12.

## 18-3. Testing

Next, let us try the memory out. Begin by moving the MAP jumper, J25, to position 1 to enable the DRAM, and then turn on the power. The speaker will probably beep and the normal HUMBUG prompt will probably appear

on the screen, but this is not enough of a test - even if the memory malfunctions, it is still possible for all this to happen since HUMBUG uses mostly the static RAM; the DRAM is used only to hold some vector addresses.

To make sure the memory works we need to do some more tests. First, use the MT (memory test) command of HUMBUG to run a quick memory check. If you have installed all 32 DRAMs, then the correct command is

```
MT FROM ADDRESS 0 TO FFFFF
```

which checks out 1 megabyte of locations from address \$0000 to \$FFFFF. If you have installed only 16 DRAMs, then the correct command is

```
MT FROM ADDRESS 0 TO 7FFFF
```

which checks out only the first 512K of memory. In each case, press the space bar after the zero and after the last F to tell HUMBUG that you have finished typing the address. Either way, HUMBUG should print out a + sign and then its normal \* prompt if the memory test passes.

Although the MT memory test of HUMBUG is fast, it is not extremely thorough. It merely goes through every location of memory and tries to set and reset every bit, one at a time. It then reads the bit back and tests it. But since it reads the bit back immediately after it sets it, the DRAM may seem to work even if the refresh circuitry is bad. We therefore need a more thorough test to check for a more long-term memory.

There are two ways to do this. One method involves copying the HUMBUG ROM into DRAM, waiting a minute or so, and then doing a memory compare to check the two against each other. Since the HUMBUG ROM is less than 20K in size, it is not big enough to fill up all of DRAM for a thorough test. Nevertheless, we can do a rough check by typing in the following two commands:

```
MO ENTER OLD ADDRESSES: FROM F80000 TO F84000
  ENTER NEW ADDRESS: 1000
MC REGION 1: FROM F80000 TO F84000
  REGION 2: 1000
```

The MO command moves the contents of ROM locations \$F80000 through \$F84000 down into RAM locations \$1000 and up (do not store anything into locations 0 through \$400, as these are used to hold other vector addresses.). The MC command, which should be done a minute or two later, does a memory comparison of the same two areas of memory. If the two sets of data do not match, then HUMBUG will display the addresses where a difference was found.

Some other possible tests are to use the FM (fill memory) command to fill all of memory with zeroes, and then the CS (checksum) command to check that the sum of all these zeroes is 00000000. Another way is to move whatever random data is in the bottom 512K into the top 512K, and then make sure that the checksum of the bottom 512K is the same as the top 512K. If you have plenty of time, you could also use the ROM-based Basic program to POKE consecutive numbers into memory locations, and then come back later, read them back with PEEK, and check against what was stored. There are many possibilities.

If all seems to work, then skip ahead to the next chapter; once we get to boot the SK\*DOS disk operating system, there will be plenty of opportunity to test out the memory more thoroughly.

## 18-4. In Case Of Difficulty

Defective DRAM circuitry can show itself in many ways - the computer can be totally dead, or it might come to life but be unreliable, or it might appear to work but simply fails memory tests. The specific troubleshooting procedures vary, depending on symptoms. The most likely symptoms are:

1. The computer is completely dead. Move the J25 jumper back to position 2 to disable DRAM and switch back to fully static RAM operation. If the computer is still dead, you have probably introduced a short circuit while soldering some of the memory sockets.

2. The speaker beeps, but nothing appears on the screen, or only the "please press enter" message appears on the monitor (unless you are using only a serial terminal), and at some point the HALT LED goes on. These symptoms generally indicate that at least part of the DRAM circuitry is working; the first step is to interchange the DRAM ICs, swapping each IC in an upper or even group with those in a lower or odd group. If the symptoms change then this generally indicates a defective DRAM IC. If, on the other hand, the symptoms stay the same, then the problem is probably elsewhere.

If an oscilloscope is available, then place J25 into position 2 to disable CPU accesses to DRAM, but keep the refreshing circuits going. Now check for the following, and trace signals if any of these appear wrong:

- (a) An 80 kHz signal at U46 pin 1
- (b) The frequency of each successive output from U46 should be half of the preceding output. For example, pin 6 should be at 40 kHz, pin 5 at 20 kHz, etc. (see Fig. 18-3).
- (c) There should be thin, barely-visible (depending on the oscilloscope) negative-going pulses at U49 pin 8 and at U66 pin 8.
- (d) There should be thin, barely visible positive-going pulses at U52 pins 8 and 12.
- (e) U49a should stay set at all times so pin 6 should always be high.

3. If the computer works, but fails a memory test, try to analyze the MT test printout to determine where in memory the problem is. For example, if errors occur in locations \$80000 and above, then the problem is only in the upper 512K of memory. If the problem occurs in a small region of memory, then the problem is likely to be only in a single IC.

Another way to narrow down defective DRAM problems is to use the ME command to store a number into memory at one of the locations flagged as defective by the MT test, and then read it back to see whether it was stored properly. For example, suppose the MT test indicates errors in odd locations between 4001 and 4FFF. This would indicate that an error is occurring in the odd or lower memory group in the first 512K of memory. Use the ME command to store the number \$FF in location 4001, one of the

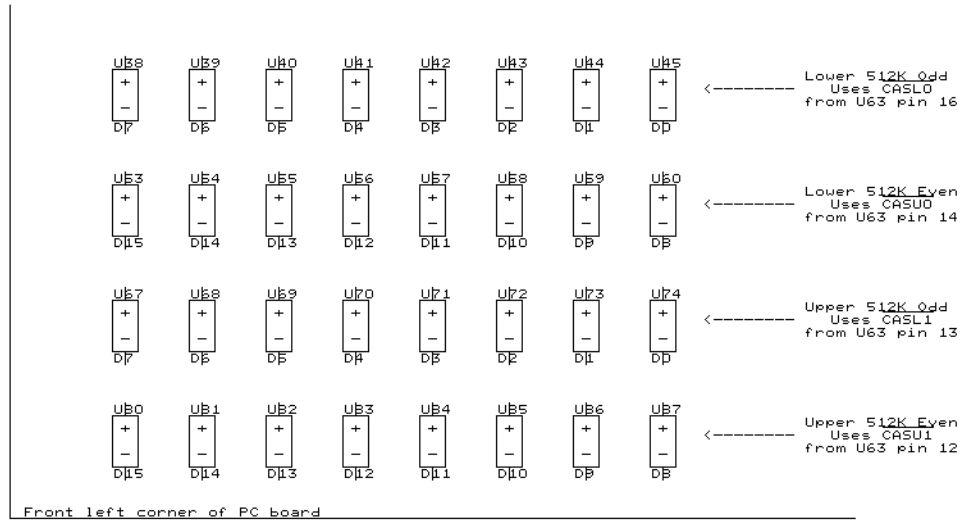


Fig. 18-5. Physical layout of the 32 RAM ICs.

defective locations. If you then read back SFB, for example, compare the bit patterns for FF (11111111) and FB (11111011). Since there is a difference in the third bit from the right (bit D2), you can then use Fig. 18-5 to identify the correct IC - it is U43, since this IC is connected to D2 in the lower 512K of memory. (Note that the bits in even locations are numbered from D15 on the left to D8 on the right, whereas the bits in odd locations are numbered from D7 on the left to D0 on the right.)

