
Chapter 7

68000 Operation In An Open Loop

Now that we have functioning clock and reset circuits, and are familiar with some of the pins on the 68000, it is time to get it to do something. Normally, you need quite a bit of external hardware to get any microprocessor running, but there is a way of fooling the 68000 into thinking there is an entire computer connected even though there is almost nothing there.

7-1. Discussion

Fig. 7-1 shows how we are going to get the 68000 to work with the minimum of external hardware.

In order to minimize the amount of extra wiring we have to do, we will take advantage of circuitry already on the printed circuit board which will be needed later anyway. For example, the $\overline{\text{RESET}}$, $\overline{\text{HALT}}$, and CLK signals are already connected on the printed circuit board, and we don't need to do anything to them at all.

U37b is a NAND gate which generates a low $\overline{\text{VPA}}$ when FC0, FC1, and FC2 are all high, but keeps $\overline{\text{VPA}}$ high at all other times. As explained in Chapter 6, it is used during interrupt processing to tell the 68000 to use auto-vectoring; for now, it will simply keep $\overline{\text{VPA}}$ high.

U89 is the priority encoder IC, also used only during interrupts. For now, however, the seven resistors in R19 are keeping all of the $\overline{\text{IRQ}}$ lines high; since they are active low, this negates them. U89 therefore sees no interrupt request, and so it sends the number 000 to the 68000, telling it there are no interrupt requests. We could have achieved the same thing by grounding the three $\overline{\text{IPL}}$ pins of the 68000 and connecting $\overline{\text{VPA}}$ high, but we can save ourselves that job by installing R19, U89, and U37, which we would have had to do eventually anyway.

Two of the 68000's inputs are already connected to a high to negate $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$, and several pins (including the entire address bus) are la-

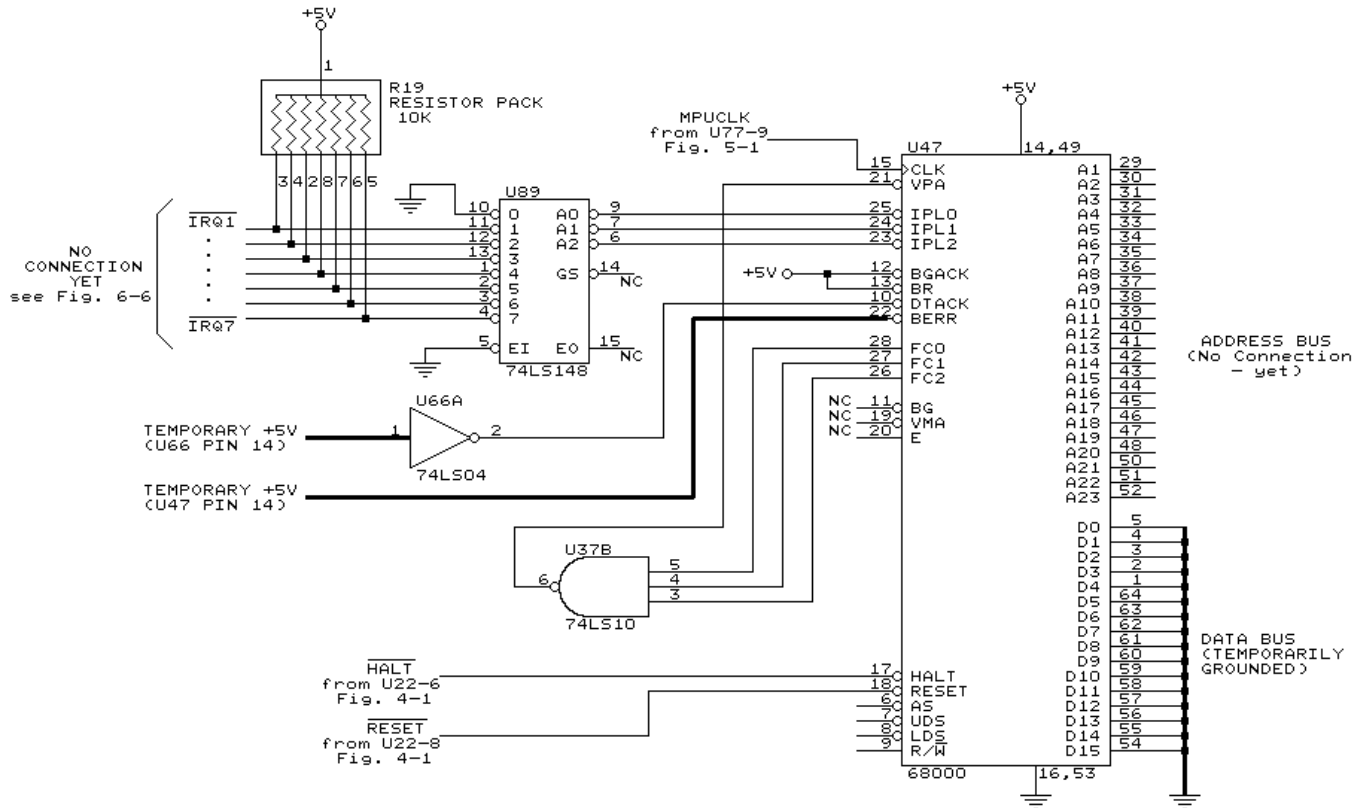


Fig. 7-1. Test circuit for the 68000.

belled NC to show that they do not, as yet, go anywhere. We need do nothing with these as yet.

That leaves the three sets of connections shown in heavy black on Fig. 7-1. First, BERR must be held high (negated) so the 68000 doesn't think a bus error has occurred; this is easily done by installing a short jumper between pins 14 and 22 of the 68000, since pin 14 is at +5 volts.

Although DTACK will normally carry a meaningful signal, for now we want to ground it to make the 68000 think that all is well on the outside. But since inverter U66 was already previously installed, we instead connect U66a input to +5 volts with a jumper from U66 pin 1 to pin 14.

Finally, when the 68000 is running, it wants to keep fetching instructions and addresses out of memory; we have to provide it with some meaningful data so it can keep going. We do so by grounding all 16 lines of the data bus. In this way, every time the 68000 tries to read anything out of memory, it will get back the number 0000. As it turns out, there is a valid 68000 machine language instruction which says "OR a 0 to register D0", abbreviated OR.B #0,D0, which consists of four bytes of 00.

Though this OR instruction does absolutely nothing, the 68000 thinks all 16 megabytes of its memory are full of nothing but 4 million OR instructions, and so it starts doing them one after another. When it gets to

the top of memory at \$FFFFFF, it simply starts all over again from \$000000.

7-2. Construction

Now let's wire up the required components and see what happens. Install the following:

- 64 pin socket for U47 (but don't plug in the 68000 yet)
- U37 74LS10 triple 3-input NAND, and its socket
- U89 74LS148 priority encoder, and its socket
- R19 10K single-in-line package. Its pin 1, identified by a white line, points toward J25
- C14 and C48 0.1 μ F disc capacitors
- A short wire jumper from U47-14 to U47-22 to negate BERR.
- A short wire jumper from U66-1 to U66-14 to assert DTACK. Both of these jumpers will be removed later, so do it neatly and in a way which is easily unsoldered later.

Grounding the entire data bus can be messy, so we will do it another way. The data bus is connected to the two EPROM sockets (U20 and U27) and the two static RAM sockets (U21 and U28) as shown in Fig. 7-2. Start by installing a 28-pin socket at U27 and a 24-pin socket at U21.

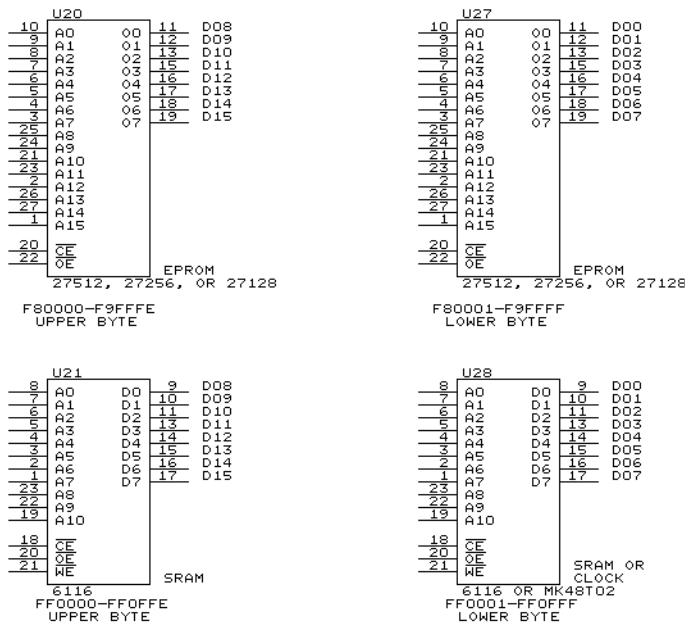


Fig. 7-2. EPROM and SRAM data bus wiring.

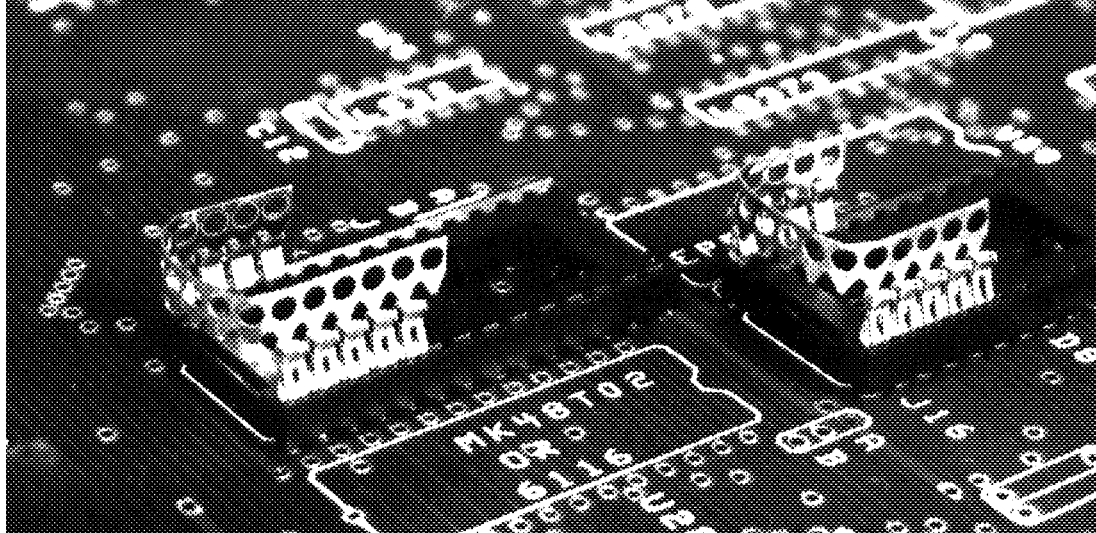


Fig. 7-3. Grounding the data bus with Molex pins.

On U27, data bus bits D0 through D7 go to pins 11-13 and pins 15-19, while pin 14 (not shown) is conveniently grounded. Thus we need to short together pins 11 through 19. In the same way, on U21, data bus bits D8 through D15 go to pins 9-11 and pins 13-17, while pin 12 (not shown) is conveniently grounded. Thus we need to short together pins 9 through 17. On both ICs, these are the bottom four pins on the left and the bottom five pins on the right. Instead of soldering, we take a strip of Molex Soldercon pins and insert them into the sockets as shown in Fig. 7-3, with four pins on the left, five pins on the right, and a section of six or so pins bent in a U shape down below. These Molex pins are normally sold as an inexpensive substitute for sockets; they consist of individual clips joined by a perforated carrier strip which would normally be broken off after soldering. In our case, we insert the thin pins (which would normally be soldered to a board) into the socket, and use the entire strip as one big short circuit.

7-3. Testing

Finally, plug in the 68000 (be careful not to bend pins) and turn on the power. Connect your LED probe to pin 52 of the 68000, which is address line A23; if all is well, the LED will light for about 2 seconds, go off for about 2 seconds, and so on. It may operate faster if your SK68K clock is faster than 8 MHz.

If the LED does not flash at the 2 seconds on, 2 seconds off rate, recheck the signal at every pin of the 68000. Look especially for the low on \overline{DTACK} , lows on all data lines, a high on \overline{BERR} , \overline{HALT} , and \overline{RESET} , and the clock signal (LED slightly less than full brightness) on CLK.

If the LED flashes as expected, all is probably well. What's happening is that the 68000 is racing through memory (or what it thinks is memory, all 16 megabytes worth), executing OR instructions at its maximum speed, one instruction every microsecond. One complete run through 4 million instructions (4,194,304 instructions, to be exact, one quarter of 16 megabytes or 16,777,216) therefore takes a bit over 4 seconds.

During that time, the address bus is counting off the addresses where the 68000 thinks those instructions are coming from. If you look at some small binary numbers such as 000, 001, 010, 011, 100, 101, 110, 111 etc., you see that the right-most bit alternates 0, 1, 0, 1, 0, 1, The second bit also alternates, but slower: 0, 0, 1, 1, 0, 0, ...; the third bit is slower still: 0, 0, 0, 0, 1, 1, 1, 1, ..., and so on. Exactly the same thing occurs on the address bus: A1 alternates back and forth very rapidly, A2 goes half as fast as A1, A3 is half as fast as A2, and so on, all the way up to A23 which alternates from 0 to 1 and back to 0 once every four seconds.

Fig. 7-4 shows some of the waveforms on the upper bits of the address bus. When the LED probe is connected to A23, it flashes on and off once every four seconds. If it is connected to A22, it repeats once every two seconds; on A21 it repeats once per second. As we go down to A20, A19, and so on, it flashes faster and faster, until at A16 (pin 44) it flashes so fast (about 32 times per second) that we can barely see it flicker; A15, flashing about 64 times per second, looks absolutely steady (though not at full brightness).

As a final check, if you have access to an oscilloscope or frequency counter, check each address line to make sure that its frequency is half that of its higher neighbor. This makes sure that there are no accidental shorts between adjacent lines of the address bus. (If using a frequency counter, keep in mind that the actual waveforms are not nearly as precise as the idealized ones of Fig. 7-4; some counters may have difficulty properly counting the frequency of such a square wave.)

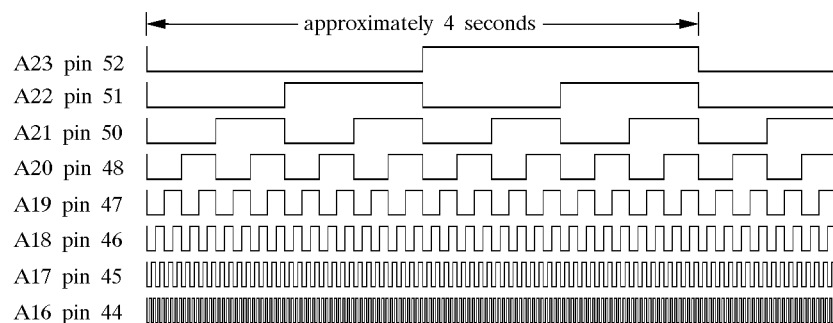


Fig. 7-4. Address bus waveforms during testing.

