

---

# Chapter 11

---

## The $\overline{\text{DTACK}}$ Circuit

Up until now, we have been generating an artificial  $\overline{\text{DTACK}}$  with a temporary jumper; it's time to substitute a real  $\overline{\text{DTACK}}$  circuit and see how it works.

### 11-1. Discussion

Ideally, each device, whether memory or I/O, should generate its own  $\overline{\text{DTACK}}$  when it has finished an operation. In this way, the 68000 would know right away that it is time to continue to the next step. That is practical in some cases, such as dynamic RAM or some I/O devices, but in others it may be necessary to build a timer which just waits a certain time and then assumes that all is well. Thus the SK-68K system takes both approaches.

Fig. 11-1 shows the heart of the  $\overline{\text{DTACK}}$  circuit. This circuit has eleven inputs, of which two (AS and CLK8) are used strictly for timing.

Five of the inputs come directly from the address decoder in Fig. 10-1. Three of these go to U37a:  $\overline{\text{I/O4}}$  goes low whenever the address decoder enables the WD1772 floppy disk controller; the EPROM  $\overline{\text{CE}}$  signal goes low when the EPROM is selected, and  $\overline{\text{SRAM}}$  goes low when the address decoder selects the static RAM. Whenever any of these three goes low, U37a provides a high output to U33. Note that U37a is a three-input "positive NAND" gate, but in this application it ORs three active-low signals and provides an active high signal whenever any of the inputs is asserted (low).

U33 is a quad D-type flip-flop, wired exactly the same as U76 in Fig. 9-1, so this time it is shown as a single block. It is clocked by CLK8, the 8 MHz clock signal, but most of the time the output of U37a is low, and so its four flip-flops are normally forced reset. That makes its  $\overline{3Q}$  output (the  $\overline{Q}$  output of the third flip-flop) normally high.

Now suppose the address decoder selects either the floppy disk controller, the EPROM, or the static RAM. The output of U37a then goes high,

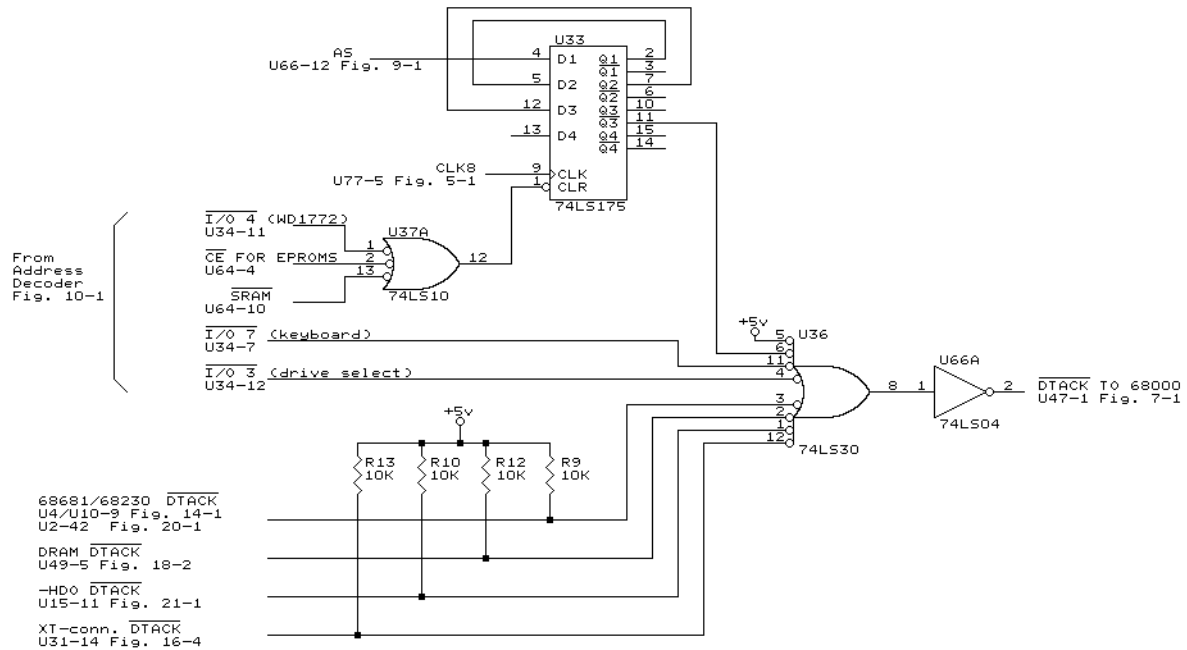


Fig. 11-1. DTACK generator circuit.

releasing the CLR signal on U33 and allowing it to start shifting. Since AS is high at this point, this high starts shifting through the flip-flops. After three CLK8 clock pulses, the 3Q output goes low. In other words, U33 acts as a delay, sending a low pulse to U36. Note that it is clocked by CLK8, not by MPUCLK. Even if you speed up the computer by using a faster MPUCLK, the delay in U33 will not change.

Now let's look at U36, an 8-input "positive NAND" 74LS30 which is used here as a "negative OR with a positive output" or, rather, to OR seven active low signals and produce an active high output (only seven inputs are needed, so pin 5 is permanently negated by being connected to +5 volts). One of these seven inputs comes from U33; one comes directly from the keyboard select line, I/O7, in the address decoder; and another comes from the drive-select enable line, I/O3, also in the address decoder. (The keyboard circuit and floppy disk drive select circuits are very fast and so I/O7 and I/O3, their select signals, immediately generate a DTACK without any other delay.)

The other four inputs to U36 come from other parts of the computer which we have not yet built or discussed. Each of these other parts generates its own DTACK when it is finished with a data transfer, and all of these are also ORed in U36. For now, however, we need resistors R9, R10, R12, and R13 to keep these four inputs high and prevent any undesired DTACKs from being generated.

U36 therefore acts as a large OR gate - whenever it receives a low pulse on any of the seven inputs, it sends a high to U66, which then finally asserts DTACK by sending a low to the 68000.

## 11-2. Construction

To construct this DTACK circuit, **REMOVE** the jumper between pins 1 and 14 of U66, which we have used until now to generate a fake DTACK. Then install the following components:

U33	74LS175 and its socket
U36	74LS30 and its socket
R9, R10, R12, R13	10K 1/4-watt resistors

U37 and U66 have already been previously installed; the J25 jumper should still be in position 1 from our previous step; and at this point you should still have the Molex pins inserted in the U21 and U27 sockets, but no other extra jumpers.

## 11-3. Testing

Now turn on the power again and let's see what happens.

Nothing! Well, of course not. The problem is that the 68000 is still trying to execute 4 million OR instructions; it is getting those instructions from our Molex pins all right, but it is not getting DTACK. Hence the BERR bus error circuit is timing out and halting the whole thing. You probably noticed that the HALT LED stays on and never even flickers.

Now move the J25 jumper from position 1 to position 2 and try again; you will see a slight flicker on the HALT LED about a second after you turn on the power (or force a reset by shorting J23), but it still lights.

Since the Molex shorting pins on U21 and U27 put all zeroes on the data bus, the start address which the 68000 gets just after a reset is also all zeroes. It therefore starts to execute a program which it thinks starts at address 000000. With J25 in position 1, low memory is supposed to be dynamic RAM; since there isn't any, there is no DRAM DTACK coming into U36; hence the 68000 quits almost immediately with a bus error. With J25 in position 2, however, the EPROM is supposed to be mapped into low memory. There isn't any EPROM, of course, but the address decoder and DTACK generator don't know that; hence they generate DTACK as if the EPROM were there. The 68000, therefore, starts to execute its OR program until it passes the address where EPROM would normally end; then DTACK suddenly disappears and the system goes into HALT as a bus error appears.

