

Chapter 13

The Magic Moment: First Signs Of Life

At this point, we can finally get the computer to do something useful - we call it the Magic Moment when it shows the very first signs of Life.

13-1. Discussion

When you turn on the power (or short the reset pins at J23), the HUMBUG monitor program in the computer's ROM tries to initialize the input and output ports, makes a list of what options you have installed, and emits a "beep-boop" sound from the speaker.

The MC68681 DUART at U10, along with the 3.6864 MHz oscillator at U3, is used mainly for the serial ports. But the DUART also has an internal counter and timer, which is used to generate the tones for the speaker. This

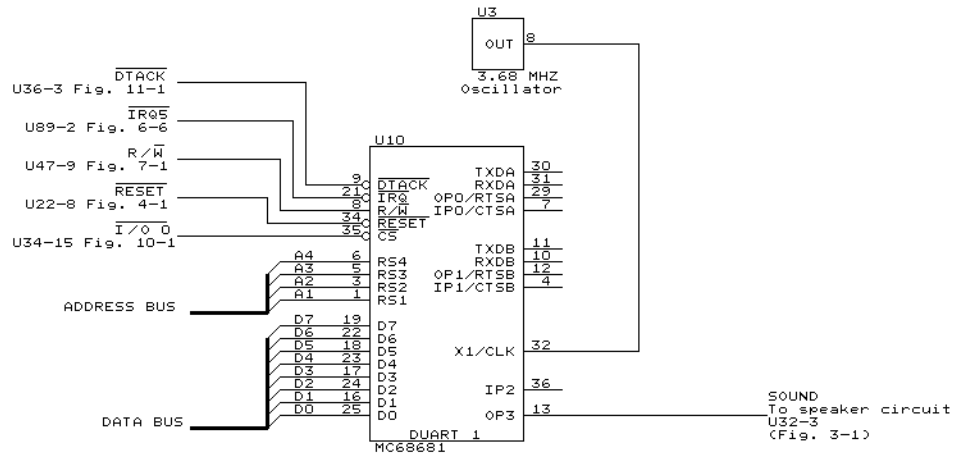


Fig13-1. DUART1 sound generator.

circuitry is shown in Fig. 13-1, though only one small wire - the connection to pin 13 of U10 - is dedicated to the speaker circuit.

The main IC in this circuit is U10, the DUART. This is a *Dual Asynchronous Receiver-Transmitter* which handles the conversions between the parallel data as it is sent along the data bus, and the serial data as it travels along the serial link to the terminal or printer. Though it carries the "UART" letters in its name, it is much more than the traditional UART, which only handles the conversion to and from serial data. Not only does the DUART contain two internal receiver/transmitter ports, but it also contains a multi-function 6-bit parallel input port, an 8-bit output port, and a 16-bit programmable timer/counter. We will discuss the other capabilities of the DUART in upcoming chapters. The DUART does use bit 3 of the output port (pin 13, called OP3) and the programmable counter/timer to drive the speaker circuitry, as shown in Fig. 13-1.

The connections to the rest of the computer (on the left of U10) are fairly straightforward. The clock signal comes from U3, a 3.6864 MHz oscillator module. The \overline{DTACK} output connects directly to the DTACK gate, U36; the \overline{IRQ} output connects to $\overline{IRQ5}$, the level 5 IRQ input; $\overline{R/W}$ and \overline{RESET} connect to the same points in the rest of the system, and \overline{CS} , the chip select input, connects to the $\overline{I/O0}$ point (U34-15). As we have already discovered, this address decoder output goes low for addresses in the range from \$FE0000 through \$FE003F.

Finally, data bus pins D0 through D7 connect to the lower eight bits of the data bus, while the four register select lines, RS1 through RS4, connect to A1 through A4, respectively. Since there are four register select lines - four address lines - the DUART occupies 16 addresses in the 68000's memory space. But since only the lower eight bits of the data bus are involved, these sixteen addresses are all odd, so the DUART is addressed at \$FE0001, \$FE0003, \$FE0005, and so on, up through \$FE001F.

Internally, a DUART contains almost two dozen registers, some of which can be read by the 68000, and some of which can be written into by the 68000. It takes Motorola about 70 pages to explain the DUART in their MC68681 data 'sheet' (Motorola publication ADI-988, available from Motorola distributors or from Motorola Semiconductor Products, 3501 Ed Bluestein Blvd., Austin TX 78721), so there is no way to do it justice here. We strongly recommend that you obtain that manual.

13-2. Construction

Install the following parts:

U3	3.6864 MHz oscillator module (possibly in a special socket)
U10	MC68681 DUART and its socket

Then connect a speaker to pins 1 and 4 of J18. The small speaker which comes with XT or AT clone cabinets is best; any other speaker or earphone will do, but choose a cheap one. The circuit puts a constant DC current

through it which, though it won't harm the speaker, will certainly not do much good to an expensive hi-fi speaker system.

13-3. Testing

Now turn on the power; about a second later, the HALT LED should go off, and another second later you should hear a beep-boop from the speaker. This is a signal from the HUMBUG program that it is up and running, and is the first big indication that the computer is really capable of running a real program - even the slightest problem would prevent HUMBUG from sounding this dual tone.

Now that a real program is running with no errors, the signals in the address decoder will change slightly from before. HUMBUG is currently running a loop which continuously checks for the presence of a keyboard; this loop runs entirely in EPROM and calls the first DUART, so you will see pulses on the EPROM chip select line, U64a pin 4; the I/O line on U64b pin 10; and the I/O0 line, U34 pin 15. You will no longer see pulses on the SRAM chip select line.

13-4. In Case Of Difficulty...

Even if your system is working properly and emits the beep-boop sound from the speaker, read the following material so as to get a good idea of what the system is doing and how one would proceed to troubleshoot it if necessary.

In case of difficulty, we start with all of the obvious possibilities.

First, using a good magnifying glass, carefully examine all solder joints to make sure all connections are soldered, but no shorts exist between adjacent lands on the board. Try to check that all IC socket pins really go through the board to the bottom side - sometimes a pin bends under the socket and never makes it through the board. Make sure that all IC's point in the same direction - pin 1 toward the left rear corner - and that each socket contains the correct IC. Check also that all capacitors and resistors are in their correct places. It might also be a good idea to check power supply voltages.

While on the subject of power supply voltages, note that the wide copper strip going all around the edge along the bottom side of the board is ground, but the edge conductor on the top of the board is +5 volts. Hence if you mount the board with metal hardware, you will short +5 volts to ground. Moreover, it is possible to short out the power supply by clipping a scope or meter ground lead to the edge of the board. Be careful in this regard.

Next, check that all the correct components are installed, but no others. Extra oscillators, resistors and capacitors can stay, but any extra ICs should be unplugged during troubleshooting, as they can cause undesirable results.

Check also all jumpers and other connections. The speaker should be connected to the two outer pins of J18; J25 should connect the center pin to

pin 2; J24 should connect the center pin to pin 1; J19 and J20 should be set according to the table in Fig. 12-1.

If all these simple tests reveal no problems, then it's time for more drastic action. At this point, there are two ways to go - the brute-force method is to go back to Chapter and then try to retrace our steps. But there is another, more elegant method which lets us use the 68000 to help us debug. Let's describe the latter.

Let's review how the 68000 works. When it wants to access memory or I/O, the 68000 outputs an address on the address bus, along with the appropriate control signals on \overline{UDS} , \overline{LDS} , etc. When the memory or I/O is finished with the requested operation, it sends back a \overline{DTACK} or data transfer acknowledge signal, which tells the 68000 that the operation is finished and it can continue to the next step. If, for some reason, an operation cannot be completed within a specified time, then the bus error circuit sends a \overline{BERR} signal to the 68000, which attempts to recover. If the recovery procedure results in another bus error, then the 68000 simply halts and the HALT LED goes on. This is most likely what is happening if your HALT LED either stays on continuously, or flickers about 1 second after you turn on power or reset the system.

To proceed, disable the \overline{BERR} circuit by pulling out U76 and connecting a 10K resistor from U76 pin 14 to pin 16. This forces \overline{BERR} always high, so that a bus error will never be received. (We have soldered two Molex pins to the leads of the 10K resistor, so we can just insert it into the socket instead of U76. Don't force the leads of the resistor into the socket, as they will stretch the socket pins.)

Next, disable the \overline{DTACK} circuit by pulling out U36 and connecting a 330-ohm resistor from U36 pin 8 to pin 7. Since U66 is installed (we can't remove it because it is used in the reset circuit), this forces the \overline{DTACK} input into the 68000 high so it never receives a \overline{DTACK} .

Now turn on the power. The very first thing the 68000 tries to do upon power up is to get a stack address and a starting address from locations 000000 through 000007, which are at the very beginning of the ROM and which contain the following data:

Address	Data	
00000000	00FF0FEA	Address for the stack
00000004	00F800C0	Starting address of HUMBUG

A bit of explanation is in order. Although addresses inside the 68000 consist of 32 bits (and thus are expressed as eight hex digits in the above), the actual 68000 address bus only consists of 24 lines (counting A0 even though it only exists internally within the 68000). Thus the first two hex digits of addresses are generally 00.

Furthermore, the 32-bit address 00FF0FEA is too long to fit into a memory location which can only hold an 8-bit byte. It is therefore stored in four consecutive locations, namely locations 00000000 through 00000003. That explains why the address of the second line in the above is 00000004 - it is the next available location after location 00000003.

Finally, although we have a 4-byte address stored in four 1-byte locations, the 68000 can read or write two bytes at a time over its 16-bit data bus. It must therefore read the 4-byte address in two parts.

So what happens is this: when the 68000 tries to read the stack address from memory, it outputs address 000000 on the address bus, makes both \overline{UDS} and \overline{LDS} low to signal that it is trying to access both an upper byte and a lower byte, makes the \overline{AS} address strobe low to activate the address decoder, and makes R/\overline{W} high to indicate it is reading. The address decoder decodes address 000000, and sends a low to both EPROMs to enable them. The EPROMs, in turn, output the contents of the first two bytes (the number \$00FF, the first half of the \$00FF0FEA stack address) to the data bus, which sends this data to the 68000. The 68000 receives the data, and now waits for \overline{DTACK} so it can continue.

But we've disabled \overline{DTACK} ! Thus the microprocessor just sits there, waiting. Take a scope, logic probe, or even a meter, and look at its pins. You will see the address 000000 on the address bus, 00FF on the data bus, both \overline{UDS} and \overline{LDS} low, \overline{AS} low, and R/\overline{W} high. (It may be useful to place a label on the top of the 68000 and label each pin). You will also see lows on the \overline{CE} and \overline{OE} pins of both EPROMs, and a low on IC64a pin 4. In other words, even with fairly simple test equipment, we can check out the circuit to make sure all the right signals are there. Table 14-1 is a complete list of what you should find on every pin of the 68000 at this point - check to see that this is so, and trace the signals if not. (The table refers to lines as being either High or Low. A high should be +3 volts or more; a low should be below 0.4 volt; if you see any lines which are between those levels, look for a possible short circuit which is connecting a high to a low and producing an in-between voltage.)

(Here's an example of how to interpret strange results. Suppose all the signals on the 68000 are correct except for the sixteen data bus lines. If the data bus reads \$FF00 instead of \$00FF, are the EPROMs interchanged? If the data bus is completely different, are the EPROMs selected (lows on their \overline{OE} and \overline{CE} inputs)? If yes, are they getting the correct address? Is U64 pin 4 outputting a low? If not, is U64 getting lows on pins 1, 2, and 3? As long as the processor is totally stopped, tracing signals through is easy.)

If everything seems to be normal, the next step is to pulse \overline{DTACK} low so the 68000 will proceed to its next step. This pulse must be wide enough so the 68000 will recognize it, and yet narrow enough so that the 68000 will only go forward one step and no more. The timing is thus touchy, but we have found that taking a discharged 0.001 μ F capacitor (short its leads together first to discharge it) and momentarily connecting it from U36 pin 8 to pin 14 works quite well. Pin 8 is being held low by the 330-ohm resistor added a few paragraphs ago, but the 0.001 μ F capacitor pulls it high for about a quarter of a microsecond; this is inverted by U66a into a low \overline{DTACK} pulse.

As a result, the 68000 now tries to read the next two bytes of the stack address from memory, so the address bus should contain the address 000002, and the data bus should have the number 0FEA - the second half of the address 00FF0FEA. There is always the possibility that the pulse from the capacitor may be too wide or too narrow, or that more than one pulse may be generated, in which case you may either stay at the same step, or

may skip ahead several steps. In that case, refer to Table 14-2, which is a list of the addresses and data that should exist on the address and data bus during the first dozen or so steps.

Looking at the address and data busses in this way should make it possible to spot address or data bus shorts or opens.

Assuming these results are normal, the next step is to remove the 330-ohm resistor, replace U36, and restart the system. The computer will now start executing HUMBUG from the beginning, going at high speed through the steps we were tracing one-by-one earlier. But any time that it tries to access a memory or I/O address that either doesn't exist or that is not properly working, it will fail to get \overline{DTACK} ; since we are still forcing \overline{BERR} high with a 10K resistor, the 68000 will therefore stop and we can again look at the address bus to see where.

Since HUMBUG tries to compile a list of installed hardware, it intentionally tries to access addresses that may not yet exist on your system. In order, these are \$FE0089 (to see whether a 68230 exists), \$FE000B (to check for DUART 1), \$FE004B (to check for DUART2), and either \$C00001 or \$D60001 (to check for the existence of PC/XT-compatible connectors). Thus the first address you see on the bus should be \$FE0089 (though, of course, we cannot see A0. Instead, \overline{LDS} will be low or on, while \overline{UDS} will be high or off, to signify that the address is \$FE0089 and not \$FE0088.) If the 68000 stops with any other address on the address bus, the 68000 is trying to access some location that does not work or does not exist.

Once you see the \$FE0089 address of the 68230, you may pulse \overline{BERR} low once to get the 68000 to continue until the next bus error. This can be done with the same 0.001 μ F capacitor as before, except this time connect it between U76 pin 14 and U76 pin 8, which is ground. As before, you may skip ahead a few extra steps if the pulses introduced by the capacitor are not quite right, in which case just start over.

Note that your computer should not stop at address \$FE000B, since you have already installed DUART1 at U10. If it does, then there is something wrong with the DUART1 circuit.

PIN NO.	STATE	SIGNAL NAME	DESCRIPTION	COMMENTS
52-50 and 48-29	Low	A23-A1	Address bus	\$000000
54-61	Low	D15-D8	Data bus	00
62-64 and 1-5	High	D7-D0	Data bus	\$FF
6	Low	\overline{AS}	Address strobe	on
7	Low	\overline{UDS}	Upper data strobe	on
8	Low	\overline{LDS}	Lower data strobe	on
9	High	$\overline{R/W}$	Read/write	Read
10	High	\overline{DTACK}	Data Xfer acknowledge	Forced to off

PIN NO.	STATE	SIGNAL NAME	DESCRIPTION	COMMENTS
11	High	\overline{BG}	Bus granted	off
12	High	\overline{BGACK}	BG acknowledge	off
13	High	\overline{BR}	Bus request	off
14	High	Vcc	+5 volts	power
15	Pulses	CLK	8 MHz clock	clock pulses
16	Low	GND	Ground	
17	High	\overline{HALT}	Halt	Not halted
18	High	\overline{RESET}	Reset	Not reset
19	High	VMA	Valid Memory Address	off
20	Pulses	E	E clock	clock pulses
21	High	\overline{VPA}	Valid Peripheral addr	off
22	High	\overline{BERR}	Bus Error	Forced to off
23	High	$\overline{IPL2}$	Interrupt inputs	
24	High	$\overline{IPL1}$	"	no interrupts
25	High	$\overline{IPL0}$	"	
26	High	FC2	68000 Function Code	Currently in
27	High	FC1	"	supervisor
28	Low	FC0	"	program mode
49	High	Vcc	+5 volts	power
53	Low	GND	Ground	

ADDRESS	DATA	EXPLANATION
000000	00FF	Initial stack address = \$00FF0FEA
000002	0FEA	
000004	00F8	HUMBUG starting address = \$00F800C0
000006	00C0	
F800C0	4EF9	
F800C2	00F8	JUMP (op code 4EF9) to \$00F80126 instruction
F800C4	0126	
F80126	4239	
F80128	00FF	A CLR.B instruction
F8012A	002D	

ADDRESS	DATA	EXPLANATION
F8012C	4239	Another CLR.B instruction
F8012E	00FF	
F80130	0C85	
F80132	48F9	A MOVEM instruction
F80134	7FFF	
F80136	00FF	
F80138	0C0E	