

---

# Chapter 15

---

## The PC-compatible Keyboard

PC-compatible keyboards are very different from the average computer keyboard in that they contain quite a bit of internal intelligence (including a buffer which stores data which the computer hasn't yet received), yet they do not generate an actual ASCII code for each key press. Such keyboards require a fairly complex interface, shown in Fig. 15-1, as well as a fairly complex program to process their output.

### 15-1. Discussion

The keyboard connects to J9 via a 5-pin DIN connector which sends ground, +5 volts, and the RESET signal to the keyboard, and receives the CLOCK and DATA signals from the keyboard. Note, however, that both CLOCK and DATA are bidirectional since the SK68K can send signals back to the keyboard via the same lines.

When the computer is turned on or reset, the BRESET input to U32e-11 (which is the same as RESET but buffered by U17) is inverted into a low and sent back to the keyboard on the CLOCK line; this causes the keyboard to reset. (Those keyboards having caps-lock and num-lock lights usually light these lights during reset.) The RESET signal is also sent to the keyboard, and at the same time clears U24, a quad flip-flop IC. Meanwhile, the HUMBUG monitor program clears U23a by reading the keyboard (using I/O7 from the address decoder, Fig. 10-1). With U23a cleared, its Q output is low; this signal goes to G, the enable pin of U25, which allows it to operate. It also goes to U32a pin 1, a 7406 open collector inverter, which then open-circuits its output and lets the keyboard's DATA line swing high or low, as needed. At the same time, the Q output of U23a, which is high, goes to the IP2 input of DUART 1.

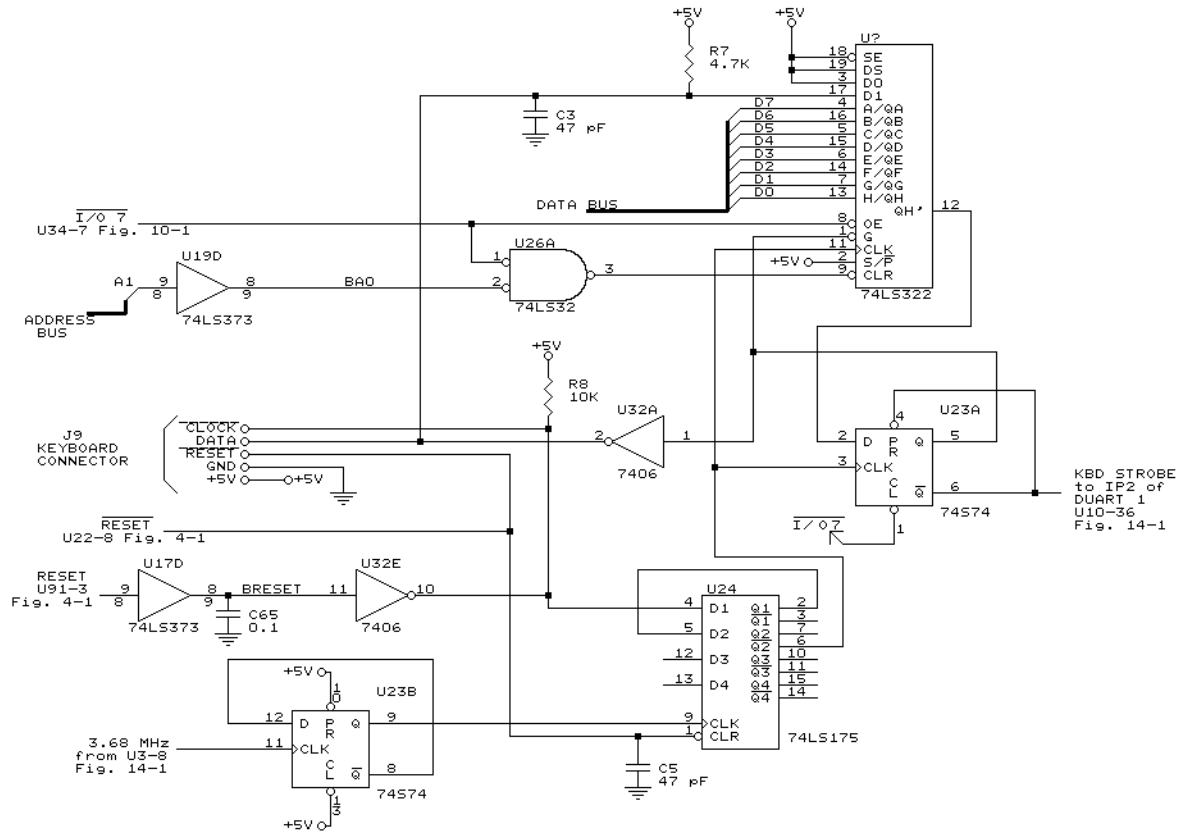


Fig. 15-1. PC-compatible keyboard interface.

Meanwhile, the 3.6864 MHz output of U3 (which is generated for the DUARTs) also goes to U23b, where it is divided by 2 to produce a 1.8432 MHz signal which clocks U24, which is wired as a shift register.

Now suppose you press (or release) one of the keyboard keys. The keyboard sends out a key number (not an ASCII code) on the DATA line as a serial code, and simultaneously starts to pulse the CLOCK line, once for each bit. The data is sent to the D1 input on U25, a 74LS322 shift register. At the same time, the CLOCK signal sends a low pulse to the 1D input of U24, whose first two flip-flops are also wired as a shift register. After two pulses of the 1.8432 MHz clock (a delay of about 1 microsecond), a high comes out the 2Q output and clocks both the U23a flip-flop and U25, the shift register. The data bit then enters the first stage of the shift register, and a moment later the CLOCK signal goes back high.

This process is repeated once for each keyboard data bit, with the data bits proceeding through U25, from QA to QB, QC, and so on, until the first bit gets to the last flip-flop, at which time a high comes out QH' and is sent to the D input of U23a. This bit happens always to be a 1, but it is immediately discarded because at the next clock pulse it is shifted right out of U25. It is there long enough, however, to make U23a set at the next clock

pulse. When this occurs, the flip-flop does four things: (a) its  $\overline{Q}$  output goes low and sends an interrupt request to DUART 1, which relays it to the 68000, (b) the same low forces U23a to stay *set* in case there are more clock pulses, (c) the Q output sends a high to the  $\overline{G}$  input of U25, which prevents it from shifting further, and (d) the same high is inverted by U32a into a low, which grounds the DATA line and prevents the keyboard from sending further data.

Assuming that DUART 1 is properly programmed, the interrupt request goes to the 68000, which then stops its normal processing and goes to a special routine called an *interrupt service routine* to accept the input from the keyboard. This routine is part of the HUMBUG monitor, which now reads a byte from location \$FE01C3; doing this read pulses the  $\overline{I/O7}$  line. This clears the U23a flip-flop, and also pulses the output enable ( $\overline{OE}$ ) pin of U25, which then sends the received data byte to the data bus to be read by the 68000. The 68000 thus gets the data byte output by the keyboard.

A few microseconds later, the *interrupt service routine* does a read from location \$FE01C1, which pulses the  $\overline{I/O7}$  line a second time, but with a difference. On the first read (from \$FE01C3), A1 (and BA0, which is the same as A1 but buffered by U19) was a 1, but on the second read (from \$FE01C1) this signal is a 0 (because C3 ends with 0011 whereas C1 ends with 0001; the second bit from the right is A1, which is now a zero.) Hence on the second read, U26a receives two low inputs at the same time, and therefore outputs a low to the  $\overline{CLR}$  input of U25, which resets it and gets it ready for the next keyboard output.

Rather than providing a specific ASCII character for each key pressed, PC-compatible keyboards send out a code each time *any* key is pressed, and a different code each time *any* key is released. Since this circuit generates an interrupt request for each such code, the interrupt service routine in HUMBUG is called each time a key is pressed or released, and has the job of keeping track of keys as well as converting these codes into ASCII. A number of books on the IBM PC describe how such a keyboard works; a particularly readable one is chapter 6 of the *Programmer's Guide to the IBM PC* by Peter Norton.

## 15-2. Construction

Now install the following components:

|                 |                                       |
|-----------------|---------------------------------------|
| R7              | 4.7K 1/4-watt resistor                |
| R8              | 10K 1/4-watt                          |
| U23             | 74S74 dual D flip-flop and its socket |
| U24             | 74LS175 quad latch and its socket     |
| U25             | 74LS322 shift register and its socket |
| U17             | 74LS373 buffer and its socket         |
| U19             | 74S373 buffer and its socket          |
| C8, C9, and C10 | 0.1 $\mu$ F disk capacitors           |
| J9              | round, black keyboard connector       |

U26, U32, and the three 47 pf capacitors have already been previously installed.

## 15-3. Testing

Now turn on the power. If you have a serial terminal connected, then testing the keyboard circuit is easy:

1. Set the terminal to 2400 baud (the default rate for HUMBUG),
2. Turn on the power,
3. Wait for the beep-boop from the speaker,
4. Press the RETURN key on the PC-compatible keyboard,
5. Press control-S, followed by the letter R, on the PC-compatible keyboard.

You can now use this keyboard for input, but will see all output on the serial terminal.

If your terminal does not run at 2400 baud, you can change the baud rate from the PC-compatible keyboard by inserting the following steps between steps 4. and 5. above:

- 4a. Type the following exactly as shown: MSFE0003
- 4b. Press the space bar once
- 4c. Type in the baud rate code listed in the table in Chapter 14 (but do not type in the dollar-sign).

For example, to change the baud rate to 9600 baud, type in MSFE0003spaceBB (press the space key - don't type in the word "space".) This sets the baud rate register of the DUART (at location \$FE0003) to the code SBB, which corresponds to 9600 baud.

Without a serial terminal, you have two choices: either use an oscilloscope or logic probe to check the various signals in the circuit, or else go on to Chapter 16, and test the keyboard after a video board is installed. The latter is probably easier.