PJRC Store

- 8051 Board, $79
- LCD 128x64 Pixel, $29
- LCD 16x2 Char, $14
- Serial Cable, $5
- 9 Volt Power, $6
- More Components...

8051 Tools

- Main Page
- **Software**
- **PAULMON Monitor**
- **Development Board**
- **Code Library**
  - Serial I/O, Polled
  - Automatic Baud Rate
  - Serial I/O, Intr.
  - Lexer
  - Random Numbers
  - Serial EEPROM
  - AMD 28F256 Flash
  - Xilinx 3000
  - **IDE Hard Drive**
  - Tiny Basic
- **89C2051 Programmer**
- **Other Resources**

Automatic Baud Rate Detection

Don't like reading docs, why not just Skip Down to the Code?

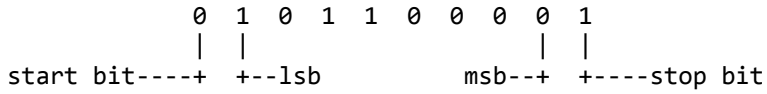
Overview

This code attempts to automatically detect the user's baud rate and initialize the 8051's built-in uart. More specifically, these steps are taken:

1. If the code was assembled with the correct baud rate, skip to step 5.
2. If we have previously detected the baud rate, skip to step 5.
3. Wait for the user to press Enter (aka Carriage Return) and count how long it takes for the bits to arrive.
4. Calculate the required timer1 reload value for the user's baud rate.
5. Store the result so that we can (hopefully) skip having to detect the baud rate if this code is called again.
6. Initialize the various special function registers.

How it works

When the user presses Enter, their terminal emulation program should send the byte 13, which will be transmitted as shown here:



Note: someday I'll draw a nice picture of this to replace the cheezy ascii art.

The code reads **P3.0**, which is the **RXD** pin, and waits for it to become low, which is assumed to be the start bit. When it becomes high again, timer1 is started in 16 bit mode. The code waits for the remaining transitions shown above. At the beginning of the stop bit, timer1 is halted. If everything went well, the 16 bit value in timer1 should indicate how many CPU cycles passed while the code waited for the eight data bits.

The 8051 uses timer1 to generate the clock for its built-in UART. Though it is possible to use timer1 in any of its modes, using the 8-bit auto-reload makes the most sense unless an extreemly slow baud rate is needed. This code only configures the timer1 in 8-bit auto-reload mode. Though this code makes configuring the built-in UART easy, it should be noted that the 8051's UART requires timer1 to generate the baud rate clock and can't be changed without affecting the UART.

The built-in UART requires 16 clock cycles (from timer1) for each bit. Because the code timed all 8 bits, the value that will be needed for initializing the timer in auto-reload mode is calculated by dividing the CPU cycle count by 128. Care must be taken to avoid a round-off error that will yield an inaccurate baud rate value. This bug appeared in PAULMON1. The code presented here fixes that bug and is identical to the automatic baud rate detection code in PAULMON2.

Available Baud Rates

Because the 8051's UART requires timer1, which is clocked by the crystal (divided by 12), the only baud rates which the 8051's hardware can produce are:

Baud = Crystal / (12 * 16 * N)

where "N" is an integer from 1 to 255. This code will select the available baud rate which is the closest to what was received. Usually, an error of about 2.5% will still manage to communicate without trouble, but much beyond that will be problematic, if it works at all. Many crystals are available which are exact multiples of the standard baud rates. Usually these crystals provide faster baud rates than simply switching to a faster crystal. For example, a 4 MHz crystal provides 1200 baud. Switching to 8 MHz only increases the maximum to 2400 baud, but switching to a 3.6864 MHz crystal will allow 19200 baud!

Here is a summary of some standard crystals and the maximum standard baud rates which should work with them:

Crystal (MHz)	Max Baud Rate	Error
1.00	300	2.12%
1.8432	9600	0.00%
2.00	300	0.79%
2.4576	300	0.78%
3.00	1200	0.16%
3.579545	300	0.23%
3.6864	19200	0.00%
4.00	1200	2.12%
4.194304	2400	1.14%
4.91520	1200	1.59%
5.00	2400	1.36%
5.0688	2400	0.00%
6.00	2400	0.16%
6.144	1200	1.23%
7.3728	38400	0.00%
8.00	2400	2.12%
10.00	4800	1.36%
10.738635	2400	1.32%
11.00	57600	0.54%
11.0592	57600	0.00%

12.00	4800	0.16%
12.288	2400	1.23%
14.31818	2400	0.23%
14.7456	38400	0.00%
15.00	38400	1.73%
16.00	4800	2.12%
18.432	19200	0.00%
20.00	9600	1.36%
22.1184	115200	0.00%
24.00	9600	0.16%
24.576	4800	1.23%
25.00	4800	0.47%
28.00	9600	1.27%
32.00	9600	2.12%

A bit of Perl code generated this table...

In applications where the speed is limited by the baud rate (such as development with PAULMON), using a crystal which provides a faster baud rate can be a big improvement, even if it is a lower frequency.

The Code

This code is available as plain text or in a zip file.
