Automatic baud rate detection for the 80C51 AN447

Author: Greg Goodhue

This note documents a method to automatically establish the correct baud rate for serial communications in many 80C51 family applications. The first character received after a program is started is used to measure the baud rate empirically.

This can eliminate the need to have setup switches whose settings are difficult to remember and all of the other headaches associated with applications that use multiple baud rates. One might assume that a reliable method of accomplishing this might be impossible without severely limiting the characters that could be recognized. The problem is in finding a timing interval that can be measured in a large number of possible characters under a wide variety of conditions.

Measuring a single bit time would be the obvious way to quickly determine what baud rate is being received. However, many ASCII characters don't have an example of a single bit time in the RS-232 pattern. For most characters, the length of the entire transmission from the start bit to the last "visible" transition will fall within certain ranges as long as some reasonable assumptions can be made about the possible baud rates (i.e. that they are standard baud rates). Moreover, many systems now use 8 data bits and no parity for ASCII transmissions. In this format, normal ASCII characters will never have the MSB set and since UARTs send data LSB first/MSB last, the program would always be able to "see" the beginning of the stop bit.

The following baud rate detection routine waits for a start bit (falling edge) on the serial input pin and then starts timer 0. At every subsequent rising edge of the serial data, the timer value is captured and saved. When the timer overflows, the last captured value will indicate the duration of the serial character from the start bit to the last 0 to 1 transition (hopefully the stop bit).

The table CmpTable contains the maximum timer measurement that is accepted for each baud rate. These values were picked such that a timed interval of only 4 data bit times (plus the start bit time) will still produce the correct baud rate.

There is an assumption in this method that anyone using it needs to be aware of. That is, that this technique depends on only one character being received during the sampling window, which has to be at least as long as a typical character at the slowest baud rate that can be accepted. Essentially this means that the data must normally come from someone typing at a keyboard.

On our PCs, we were not able to fool the program by typing two characters in quick succession. The PC function keys did present a problem because they send two characters in a tight sequence, and fooled the program into detecting the wrong baud rate. In the example program, which is designed for a 12 MHz clock, the total sample interval is about 65 milliseconds, or about twice the duration of an RS-232 character sent at 300 baud.

If parity is used, a possibility of a baud rate determination error happens when the four MSBs and the parity bit of the character received are all ones. This can happen for the lower case letters "p" through "z", plus curly brackets, vertical bar (|), tilde (~), and "delete", depending on whether the system uses odd or even parity. Note that the usual prompt characters that a user would type to get a system's attention (e.g. space, carriage return, and escape) are NOT subject to this limitation.

Because of the way this program works, the first input character that is used to detect the baud rate is lost since the UART cannot be set to the correct baud rate until after the first character has been timed. Also, most "real" programs using this technique would want to repeat the baud rate detection process if framing errors are detected at the UART during normal operation.

To calculate CmpTable values for other oscillator frequencies and baud rates, use the following equation:

$$\text{Table entry} = \frac{\text{Osc(MHz)}}{\text{Baud Rate}} \times \frac{5}{12}$$

Remember that the table entry is a two byte value, so the result of the above must be split into upper and lower bytes (easy if you have a hexadecimal calculator). It may also be possible to get the assembler to do all of the calculations for you.

The above equation was derived as follows:

$$\begin{matrix}\text{maximum}\\ \text{timer value}\\ \text{(table entry)}\end{matrix} = \frac{\text{minimum recognition time}}{\text{machine cycle time}}$$

$$\begin{matrix}\text{Minimum}\\ \text{recognition}\\ \text{time}\end{matrix} = \frac{\text{bits-to-recognize}}{\text{\#-of-bits}} \times \text{byte time}$$

Note: '#-of-bits' (the number of "visible" bits) is 9, and bits-to-recognize (the minimum # of bits to recognize) is 5 for 8-N-1 communication.

$$\text{byte time} = \frac{1}{\text{baud rate}} \times \text{\#-of-bits}$$

$$\begin{matrix}\text{machine}\\ \text{cycle time}\end{matrix} = \frac{\text{Osc frequency}}{12}$$

```
;****************************************************************************
;                     Automatic Baud Rate Detection Test
;****************************************************************************

$Title(Automatic Baud Rate Detection Test)
$Date(12–16–91)
$MOD552


;****************************************************************************
;                                 Definitions
;****************************************************************************

RX        BIT   P3.0            ;Location of serial receive pin.
CharH     DATA  30h             ;Holds high byte of frame timer result.
CharL     DATA  31h             ;Holds low byte of frame timer result.
BaudRate  DATA  32h             ;Holds final baud rate determination.
Display   EQU   P4              ;Port to display result for debug.


;****************************************************************************
;                         Reset and Interrupt Vectors
;****************************************************************************

          ORG   8000h

Start:    ACALL AutoBaud         ;Go try to get a baud rate value.
          MOV   Display,BaudRate ;Display baud rate value for debug.
          SJMP  Start


;****************************************************************************
;                                 Subroutines
;****************************************************************************

; AutoBaud Rate Detect Routine.
;   Attempts to detect baud rate from first received character, by measuring
;     the length of the character. Some characters may not work properly,
;     primarily those that end with more than 3 (4?) ones in a row.
;   Returns with ACC = baud rate pointer.

AutoBaud: MOV   TMOD,#01h       ;Initialize timer 0 (UART baud rate timer).
          MOV   TH0,#0          ;Put timer 0 in 16–bit counter mode.
          MOV   TL0,#0
          MOV   TCON,#0

          MOV   CharH,#0        ;Initialize timer result.
          MOV   CharL,#0

AB0:      JB    RX,AB0          ;Wait for serial start bit.
          SETB  TR0             ;Start timer.

AB1:      JB    TF0,AB3         ;Check for timer overflow.
          JNB   RX,AB1          ;Check for a rising edge on serial data.
          MOV   CharH,TH0       ;Capture timer value at this serial edge.
          MOV   CharL,TL0

AB2:      JB    TF0,AB3         ;Check for timer overflow.
          JB    RX,AB2          ;Check for falling edge on serial data.
          SJMP  AB1             ;Go back and repeat sampling.
```

```
AB3:      CLR   TR0             ;Maximum sample time has expired, check result.
          CLR   TF0             ;Begin by stopping timer and clearing flag.

          MOV   BaudRate,#19    ;Set up table pointers.
CmpLoop:  MOV   A,BaudRate
          MOV   DPTR,#CmpTable
          MOVC  A,@A+DPTR       ;Get a table entry for comparison.
          DEC   BaudRate
          CJNE  A,CharH,Cmp1    ;Check result range.
          SJMP  CmpLow          ;High byte table = timed value, check low byte.
Cmp1:     JC    CmpMatch        ;A match if table value is < timed value.
          DJNZ  BaudRate,CmpLoop ;Check for end of comparison table.
          SJMP  CmpMatch

CmpLow:   MOV   A,BaudRate
          MOVC  A,@A+DPTR       ;Get a table entry for comparison.
          CJNE  A,CharL,Cmp2    ;Check result range.
          SETB  C               ;Match if equal.
Cmp2:     JC    CmpMatch        ;C set if A < low byte of result.
          DJNZ  BaudRate,CmpLoop ;Check for end of comparison table.

CmpMatch: MOV   A,BaudRate      ;Comparison complete,
          CLR   C               ;  get final baud rate index,
          RRC   A
          MOV   BaudRate,A      ;  and save.
          RET

; Compare table holds timer values for the transition points of the accepted
;   baud rates. Entries are LSB, MSB. These values are for 12 MHz operation.

CmpTable:
          DB    40h,0           ;0 – out of range, value too low.
          DB    80h,0           ;1 – 38400 baud.
          DB    0,01h           ;2 – 19200 baud.
          DB    0,02h           ;3 –  9600 baud.
          DB    0,04h           ;4 –  4800 baud.
          DB    0,08h           ;5 –  2400 baud.
          DB    0,10h           ;6 –  1200 baud.
          DB    0,20h           ;7 –   600 baud.
          DB    0,40h           ;8 –   300 baud.
          DB    0,80h           ;9 – out of range, value too high.

          END
```