自动检测 80C51 串行通讯中的波特率

本文介绍一种在 80C51 串行通讯应用中自动检测波特率的方法。按照经验，程序起动后所接收到的第 1 个字符用于测量波特率。

这种方法可以不用设定难于记忆的开关，还可以免去在有关应用中使用多种不同波特率的烦恼。人们可以设想：一种可靠地实现自动波特检测的方法是可能的，它无须严格限制可被确认的字符。问题是：在各种的条件下，如何可以在大量允许出现的字符中找出波特率的定时间隔。

显然，最快捷的方法是检测一个单独位时间（single bit time），以确定接收波特率应该是多少。可是，在 RS-232 模式下，许多 ASCII 字符并不能测量出一个单独位时间。对于大多数字符来说，只要波特率存在合理波动（这里的波特率是指标准波特率），从起始位到最后一位“可见”位的数据传输周期就会在一定范围内发生变化。此外，许多系统采用 8 位数据、无奇偶校验的格式传输 ASCII 字符。在这种格式里，普通 ASCII 字节不会有 MSB 设定，并且，UART 总是先发送数据低位（LSB），后发送数据高位（MSB），我们总会看见数据的停止位。

在下面的波特率检测程序中，先等待串行通讯输入管脚的起始信号（下降沿），然后起动定时器 T0。在其后的串行数据的每一个上升沿，将定时器 T0 的数值捕获并保存。当定时器 T0 溢出时，其最后一次捕获的数值即为从串行数据起始位到最后一个上升沿（我们假设是停止位）过程所持续的时间。

CmpTable 表格列出了每一波特率的最大测量时间。这些数据是经过选择的，所以，4 个数据位时间（加上起始位时间）仍可产生正确的波特率。

使用这种方法时，必须遵守一个假设：这种技术仅取决于所接收到的一个字符，接收这个字符的波特率必须大于最低波特率。本质上来说，这意味着这个字符必须来自正常敲击键盘时所产生的字符。

在 PC 上，我们不可能快速、连续地敲击两个字符，以欺骗程序。但是，PC 的功能键具有一个问题，因为它会连续发送两个紧挨着的字符，使程序检测得到错误的波特率。在为 12MHz 时钟频率而设计的的例子程序中，其总采样时间大约为 65mS，大约可以在 RS-232 通讯中以 300bps 的速度发送两个字符。

假如使用了奇偶校验，当 4 个 MSB 以及所接收字节的奇偶校验位均这同一值时，就可能会发生错误。这类错误的发生取决于系统是使用了奇校验或偶校验，可能发生于小写的字母“p”到“z”，还有花括号（{ }）、垂直条（|）、波纹线（~），以及删除键“delete”。值得注意的是，惯常的提示符按键（如，空格键、回车键、及返回键），是**没有这些限制的**（奇数还是偶数的限制？）。

在以此方式运行程序时，如第一个字节已经过去，但串行口（UART）的波特率未能正确设置，那将造成用于检测波特率的第一个字符丢失。同样，如果在正常通讯中检测到串行口的通讯“帧”错误，绝大部分“实时”程序必须重复这一检测波特率的过程。

如需采用另外设定的晶体振荡频率、波特率，请使用下列公式计算 CmpTable 的表项目：

$$\text{表项目} = \frac{\text{振荡频率（}MHz\text{）}}{\text{波特率}} \times \frac{5}{12}$$

记住，表项目是两个字节的数值，所以上述公式的结果一定要分成高位字节及低位字节（如果采用十六进制，则容易得出高位、低位字节）。当然，也可以用汇编程序来完成所有的运算。

上述的公式是由以下得来的：

$$\text{最大定时数值（表项目）} = \frac{\text{最小认可时间}}{\text{机器周期}}$$

$$最小认可时间=\frac{最小认可位数（bits\_of\_recognize）}{“可见”位数（\#\_of\_bits)}\times字节时间$$

备注：在 8-N-1 格式的数据通讯中，‘#-of-bits’（“可见”位数）是 9，以及‘bits-to-recognize’（最小认可位数）是5。

$$字节时间=\frac{1}{波特率}\times“可见”位数（\#\_of\_bits）$$

$$机器周期=\frac{振荡频率}{12}$$

```
; ********************************************************************************
; 自动的波特率检测程序
; ********************************************************************************
$  Title(Automatic Baud Rate Detection Test)
$  Date(12-16-91)
$  MOD552
; ********************************************************************************
;  Definitions
; ********************************************************************************
RX          BIT     P3.0        ; 串行口的接收管脚
CharH       DATA    30h         ; 捕获定时器T0的高位字节
CharL       DATA    31h         ; 捕获定时器T0的低位字节
BaudRate    DATA    32h         ; 存贮最终确定的波特率
Display     EQU     P4          ; 显示结果的端口
; ********************************************************************************
;  Reset and Interrupt Vectors
; ********************************************************************************
            ORG 8000h
Start:      ACALL   AutoBaud                ; 检测波特率
            MOV     Display,    BaudRate    ; 显示波特率值
            SJMP    Start
; ********************************************************************************
;  Subroutines
; ********************************************************************************
; AutoBaud Rate Detect Routine.
; 通过测量接收第一个字符所需要的时间来确定波特率。部分接收字符可能会发生错误，
; 主要是那些以3（4？）位同样数值结束的字符。波特率指针（检测结果）保存在ACC中。
; ********************************************************************************
AutoBaud:   MOV     TMOD,   #01h        ; 初始化T0(串行口波特率定时器)
            MOV     TH0,    #0          ; 将T0 置于16位定时器模式
            MOV     TL0,    #0
            MOV     TCON,   #0
            MOV     CharH,  #0          ; 预置波特率检测结果
            MOV     CharL,  #0
AB0:        JB      RX,     AB0         ; 等待串行通讯起始
```

```
            SETB    TR0                     ; 起动定时器 T0
AB1:        JB      TF0,        AB3         ; 检查定时器是否溢出？
            JNB     RX,         AB1         ; 检测串行信号上升沿？
            MOV     CharH,      TH0         ; 在串行信号上升沿捕获定时器T0数值
            MOV     CharL,      TL0
AB2:        JB      TF0,        AB3         ; 检查定时器是否溢出？
            JB      RX,         AB2         ; 检查串行信号下降沿？
            SJMP    AB1                     ; 返回，继续采集
AB3:        CLR     TR0                     ; 最大的采集时间已经超过，检查结果
            CLR     TF0                     ; 清除定时器溢出标志
            MOV     BaudRate,   #19         ; 设置波特率表指针
CmpLoop:    MOV     A,          BaudRate
            MOV     DPTR,       #CmpTable
            MOVC    A,          @A+DPTR     ; 取一个表项目（高位字节）以进行比较
            DEC     BaudRate
            CJNE    A,    CharH,    Cmp1    ; 捕获值与表项目的高位字节相等？
            SJMP    CmpLow                  ; 高位字节相等，检查低位字节
Cmp1:       JC      CmpMatch                ; 表项目小于定时值，则符合？
            DJNZ    BaudRate,   CmpLoop     ; 未至表项目的结尾，则继续？
            SJMP    CmpMatch                ; 至比较结束
CmpLow:     MOV     A,          BaudRate
            MOVC    A,          @A+DPTR     ; 取一个表项目（低位字节）以进行比较
            CJNE    A,    CharL,    Cmp2    ; 捕获值与表项目的低位字节相等？
            SETB    C                       ; 结果相等
Cmp2:       JC      CmpMatch                ; 如果表项目＜定时值，则置位C
            DJNZ    BaudRate,   CmpLoop     ; 未至表项目的结尾，则继续？
CmpMatch:   MOV     A,          BaudRate    ; 数据比较完成
            CLR     C                       ; 产生结果（波特率索引）
            RRC     A
            MOV     BaudRate,   A           ; 保存结果
            RET
; ********************************************************************************
;  CmpTable    比较表
; ********************************************************************************
; 比较表所保持的定时值用于公认的波特率转换情况。表项目为低位（LSB）、高位（MSB）。
; 这些数据是以12MHz为基准操作。
CmpTable:   DB 40h,0                ; 0 - 超出范围，值太低
            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 - 超出范围，值太高
END
```

附： 波特率自动检测程序（通过验证）

```
            RX      BIT     P3.0        ;串行数据接收端
            CharH   EQU     30H         ;计时数据高位 TH0
            CharL   EQU     31H         ;计时数据低位 TL0
            BaudRt  EQU     32H         ;波特率计算值
;subroutine
AutoBaud:   MOV     TMOD,   #01H        ;初始化“T0”为计时器
            MOV     TH0,    #0
            MOV     TL0,    #0
            MOV     TCON,   #0
            MOV     CharH,  #0
            MOV     CharL,  #0
            JB      RX,     $           ;等待通讯开始位
            SETB    TR0
CHK1:       JBC     TF0,    CHK_END     ;若溢出，则开始计算
            JNB     RX,     $-2         ;检测串行数据上升沿
            MOV     CharH,  TH0         ;捕获“T0”计时数
            MOV     CharL,  TL0
            JBC     TF0,    CHK_END     ;若溢出，则开始计算
            JB      RX,     $-2         ;检测串行数据下降沿
            SJMP    CHK1
CHK_END:    CLR     TR0                 ;停止计数器
            MOV     DPTR,   #baudtable
            MOV     BaudRt, #19
LOOP:       MOV     A,      BaudRt      ;
            MOVC    A,      @A+DPTR     ;取表格数据（高位）
            DEC     BaudRt              ;索引地址减 1
            CJNE    A,  CharH,  CMP_1   ;检查结果范围
            SJMP    CMP_LOW
CMP_1:      JC      MATCH               ;若表中值 < 计时值，则匹配
            DJNZ    BaudRt,     LOOP
            SJMP    MATCH               ;表查完，至结束查表程序
CMP_LOW:    MOV     A,      BaudRt      ;高位相等，比较低位
            MOVC    A,      @A+DPTR
            CJNE    A,  CharL,  CMP_2
            SETB    C                   ;相等则匹配
CMP_2:      JC      MATCH               ;若低位字节 < 计时值，则匹配
            DJNZ    BaudRt,     LOOP
MATCH:      MOV     A,      BaudRt      ;转换为波特率索引值
            CLR     C
            RRC     A
            MOV     BaudRt,     A       ;保存
            RET
;波特率索引表（LSB 在前，MSB 在后,晶振为 11.0592MHz）
baudtable:  DB      03CH,00H    ;0-越限，值太小
            DB      078H,00H    ;1-波特率 38400
            DB      0F0H,00H    ;2-波特率 19200
            DB      0E0H,01H    ;3-波特率  9600
            DB      0C0H,03H    ;4-波特率  4800
            DB      080H,07H    ;5-波特率  2400
            DB      00H,00FH    ;6-波特率  1200
            DB      00H,01EH    ;7-波特率   600
            DB      00H,03CH    ;8-波特率   300
            DB      00H,078H    ;9-越限，值太大
            END
```