AN962

在 dsPIC30F 器件上实现自动波特率检测

作者： *Mike Garbutt*
Microchip Technology

介绍

目前所有 dsPIC30F 器件都拥有一个具备自动波特率检测能力的 UART 外设。UART 接收引脚（RX 引脚）上的信号能在内部传送至一个输入捕捉模块以获得输入信号边沿的时序。根据该时序，应用程序能正确设置 UART 的波特率。

当输入数据的波特率以及处理器的振荡器频率未知时，自动波特率检测是很有用的。由于 RC 振荡器经常不够精确且随时间变化会产生漂移，因此采用 RC 振荡器的系统非常适合采用自动波特率检测。

方法

自动波特率检测的方法取决于接收到的已知数据。为实现自动波特率检测，通常可使用通信协议发送特定数据。根据已接收数据的时序可计算 U1BRG 或 U2BRG 寄存器的值。上述寄存器用来设定 UART 的波特率。

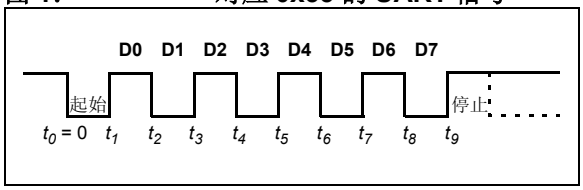
本应用笔记中的两个示例都使用输入数据 0x55（ASCII 字符 “U”）来计算波特率发生器的值。该特定数据字节提供了最大数目的脉冲边沿，因此具有最大的准确度。实际上可采用任何数据字节，此时波特率计算将会自动适应被测数据。通常，数据中出现的边沿（位状态变化）越多，则结果会越精确。

信号

UART 信号的发送顺序实行低位在先的原则。发送字节中首先是起始位（逻辑零），末尾则为停止位（逻辑 1）。发送字节通常具有 8 个数据位，但也可使用不同长度的数据位，在数据位之后可加入奇偶校验位。所有这些都将对自动波特率检测的计算造成影响。但应事先知道数据的格式。

由于存在起始位和停止位，因此存在至少两个脉冲边沿，但也可能是 10 个或更多。以数据 0x55 为例，如图 1 所示，它拥有 10 个脉冲边沿。

图 1：　　对应 0x55 的 UART 信号



时序和采样

在记录了脉冲边沿的时序并去除偏移量 t_0 后，可利用公式 1 来计算 UxBRG 寄存器的值。在附录 A 中采用线性回归算法对该公式进行了推导。计算是在最后一个脉冲边沿被记录后进行的，且应在下一个字节的起始位出现前完成以避免造成数据丢失。因此，计算过程须设定时间限制并应对其进行检查。在某些情况下，有必要使用一种具有较快执行速度的较简便计算方法。此时，同样需考虑误差检查所需的时间。

公式 1：　　UxBRG 线性回归计算

$$UxBRG = \frac{2(t_1 + 2t_2 + 3t_3 + \ldots + 9t_9) - 9(t_1 + t_2 + t_3 + \ldots + t_9)}{2640} - 1$$

误差

误差检查是一个好的编程习惯。输入信号中可能不包含期望数据 0x55。误差可能是明显的，因为边沿之间的时间较长且简便的超时即可对该问题进行检测。假使所有边沿都出现在可接受的超时周期内，这些边沿仍然有可能与期望的顺序不匹配。有几种统计方法可用来显示被测信号偏离期望信号的程度。

附录 B 的示例代码中使用了平均绝对误差。如果距离期望时间测量值偏移量的平均值超过位时间的 5%，则该数据将被丢弃。

替代方法

因为使用了所有的输入捕捉数据，线性回归方法具有极佳的准确性。但这种方法对于某些应用来说计算量太大且太慢。附录 A 还推导了一种简化的方法，该方法用两个边沿之间的时间差除以位数以计算单个位时间。

公式 2： UxBRG 的简化计算

$$UxBRG = \frac{(t_8 - t_0)}{128} - 1$$

附录 C 中的代码使用了这种方法。由于右移操作可以方便有效地实现除以 2 的幂的运算，因此公式 2 使用 8 个位时间以简化计算。附录 C 中的实例通过减少一些误差检查以获得更高速度，但代价是可靠性有所降低。

代码实例

附录 B 和附录 C 中的代码是基于输入字节 0x55（ASCII 字符 “U” ）实现自动波特率检测。这些代码是使用 MPLAB® C30 编译器进行开发的。

结构

软件具有三个主要部分：主循环，初始化和中断服务程序。

主循环

`main()` 函数具有一个执行自动波特率检测功能的无限循环。函数开头调用 `SetupAutoBaud()` 对所有外设和自动波特率检测过程所使用的中断进行初始化。随后代码处于循环等待状态直至 U1BRG 寄存器具有一个非零值，表明自动波特率检测过程已成功完成。

代码根据U1BRG中的值来计算实际波特率，并从UART发送一条报文显示计算后的波特率。波特率计算不是自动波特率检测工作所必需的，但该项工作用于演示目的。

待该文本发送后，程序重新循环并重新开始自动波特率检测过程。

初始化

`SetupAutoBaud()` 函数对UART1外设、输入捕捉1模块和 Timer 3 进行初始化以对输入数据执行自动波特率检测。

UART1启用并使能自动波特率检测功能。此时U1RX引脚上的输入串行数据信号在内部传送到输入捕捉 1 模块。

为获得最高的精度， Timer 3 设定在每一个指令周期进行递增计数。周期设置为最大值，以便在 16 位计数计满之后定时器才会出现溢出返回。

输入捕捉 1 模块设置为捕捉 Timer 3 且在输入信号的每一个边沿产生中断。输入捕捉 1 模块中断使能。

中断服务程序

共有两个中断服务程序，分别为 Timer 3 中断和输入捕捉 1 中断。

Timer 3 中断服务程序对自上一次输入捕捉事件后的定时器溢出中断次数进行计数。如果脉冲边沿之间存在一次以上的溢出，则表明自动波特率检测失败并将重新开始该过程。这种情况发生在输入信号脉冲边沿间存在较大间隔时。

输入捕捉 1 中断服务程序是自动波特率检测过程的核心部分。根据当前执行的是简化计算或复杂的回归计算，该中断服务程序将有所不同。

中断服务程序首先将前一脉冲边沿的记录时间进行保存并读入与已经检测到的当前脉冲边沿对应的新时间值。定时器溢出时计数将被复位为零以开始一个新的超时周期。

第一次捕捉中断将使能 Timer3 中断以对超时进行检查并对自动波特率检测计算变量进行初始化。

随后发生的每一次捕捉中断都用来从前一时间减去当前捕捉时间值以获得当前位的时间。该计算采用无符号整数来进行的，因此相邻捕捉间定时器是否溢出不会产生影响。

对于简便计算，位时间加入到一个累加和直至所有 8 个位时间都已被累积在一起。

对于回归计算，位时间加入到前一次记录的时间并存储到时间数组的下一个元素中。这为回归计算以及以后的误差检查提供了数据点。回归计算所需的累加运算也是在中断服务程序中完成的。

当最后一个（第十）捕捉中断发生后，中断服务程序随后禁止全部两个中断，结束自动波特率检测计算并使能使用新波特率的 UART。

对于简化计算，除了超时检查外将不进行误差检查，且 U1BRG 寄存器的值直接来自 8 个位时间的累加和。在除以 128 前，该数通过加上 64 进行舍入。这将在截取前有效加入 ½。除以 128 的操作是通过进行 7 次移位来实现的。

对于回归计算，将计算回归线的斜率和 Y 截距。这将用来计算期望时间值，该值将从用于误差检查的实际时间测量值中减去。如果误差大于 5%，自动波特率检测过程将重新开始。该门限值可以根据不同的准确度要求进行改动。最后，根据回归线的斜率计算 U1BRG 值。

代码的使用

本示例代码是使用运行于 29.5MIPS 的 dsPIC30F6014 器件在 dsPICDEM™1.1 演示板上进行开发和测试的。时序允许将代码应用于低至 600 的任何标准波特率。可通过以下步骤对示例代码进行测试：

- 使用标准的 RS232 电缆连接 PC 的 COM 口和 dsPICDEM 1.1 控制板上标有 “PORT B”的连接器。
- 在 PC 上运行终端程序，如超级终端，应确保终端程序使用了正确的 COM 口。
- 编译代码，将代码烧写到 dsPIC® 器件中，并运行代码。
- 在终端程序中键入 “U”。
- dsPICDEM1.1 演示板将以 “Baud rate: xxxx”文本作出响应，其中 xxxx 是使用的波特率。
- 改变波特率并在终端程序中再次键入 “U”。
- dsPICDEM1.1 演示板将响应新的波特率。

使用的资源

除 UART 外，自动波特率检测代码还将用到程序存储器和数据存储器，一个输入捕捉模块以及一个定时器。自动波特率检测过程中用到的 RAM 很少且在执行完毕之后定时器和输入捕捉模块还可用于其它用途。程序存储器的使用取决于计算的复杂程度。表 1 中显示的存储器是由于应用中加入自动波特率检测代码而额外使用的。所有 MPLABC30 应用程序都有最少量的代码来处理启动、初始化等工作，不过这并没有包括在图中。

调整和改进

提供的示例代码给出了两种方法对输入数据字节 0x55（ASCII 字符“U”）执行自动波特率检测。这些方法可应用于任何已知的输入数据。所有方法都通过确定输入脉冲沿的时序来确定单个位周期。可以使用不同程度的分析和误差检查措施以提高可靠性或运行速度。

对于未知数据也可使用自动波特率检测方式，但难以确定单个位时间并难以将起始和停止位与数据位进行区分。理想的情况下自动波特率检测过程应利用数据的相关信息来简化计算。

自动波特率检测过程是利用中断在后台执行的。 dsPIC 器件灵活的中断优先级结构允许自动波特率检测中断配置为对应用的其余部分影响最小。用于自动波特率检测的定时器和输入捕捉资源在自动波特率检测计算完毕之后还可用于应用的其他部分。

备用中断矢量表可用于自动波特率检测中断，允许主应用程序拥有自己单独的输入捕捉和定时器中断。

两个 UART 都具有自动波特率检测能力。注意，由于 UART1 使用输入捕捉 1 模块而 UART2 使用输入捕捉模块 2，因此两个 UART 都可同时执行自动波特率检测。

结论

内置的自动波特率检测功能使得未知波特率的 UART 的配置变得简便。由于该过程可在中断控制下在后台完成，因此其对应用的其余部分影响甚小。代码可根据用户需求，使用一种简便的计算方法来获得较高的执行速度，或使用一种具有完备误差检查功能且较为复杂的计算方法以实现更佳的可靠性。

参考文献

dsPIC30F 系列参考手册（DS70046C_CN）

dsPICDEM™ 1.1 Development Board User's Guide（DS70099）

MPLAB® C30 C 编译器用户指南（DS51284C_CN）

表 1： 自动波特率检测代码使用的资源

资源	简便计算	回归计算
程序存储器	321 字节	834 字节
数据存储器	14 字节	58 字节
I/O 引脚	无额外的 I/O	无额外的 I/O
外设	输入捕捉 1， Timer3	输入捕捉 1， Timer3
最大中断执行时间	49TCY	1486TCY*

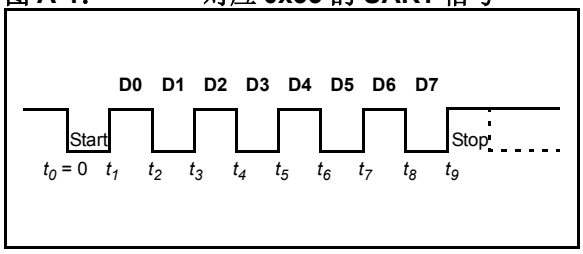
* 如果没有平均绝对误差计算，为 618TCY

附录 A： 计算

回归计算

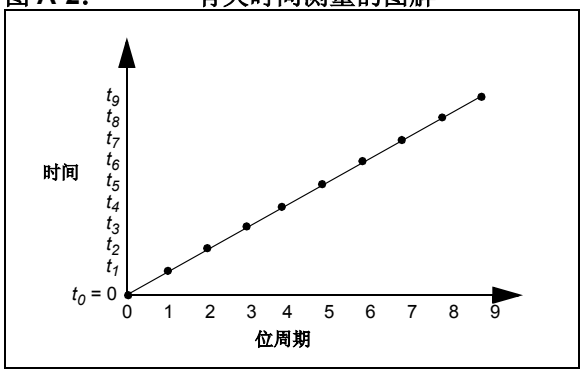
输入捕捉模块用来获得如图 A-1 中所示输入信号边沿的时序。对于一个为 0x55 的数据字节，如图 A-1 所示，采集了以 t_0=0 为起始的十个时间值 t_0,t_1, t_2… t_9 。

图 A-1： 对应 0x55 的 UART 信号



如果输入信号是正确的，时间点应等间距相隔一个位周期，将获得如图 A-2 所示的直线图。

图 A-2： 有关时间测量的图解



图中的直线斜率为每个数据位的测量时间周期。通过将任何相邻两个时间值相减可以计算出该斜率。如要获得更高的准确性，可将最后一次时间 t_9 减去第一次时间 t_0 再除以位周期的个数 9。这个方法虽然也是只用了两个时间测量值进行计算，但对于大多数应用场合已经足够准确了。

使用线性回归方法可获得图中直线斜率准确度最高的表达式。作为一项统计技术，线性回归方法可以找到通过一组点的最佳直线。公式 A-1 给出了经过 *n* 个点 $(x_0,y_0),(x_1,y_1),(x_2,y_2)\ldots(x_n,y_n)$的直线的斜率*m*的线性回归计算公式。

公式 A-1： 线性回归公式

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$

用 0,1, 2… 9 替换该公式中的 x，并用 t_0, t_1, t_2… t_9 替换该公式中的 y。由于所有 x 值都是已知的，且 t_0= 0， n = 10，所以公式可简化为公式 A-2。

简化的公式需要进行十次乘法和一次除法操作，但这对于 dsPIC30F 器件是易于实现的。

公式 A-2： 每一位周期的时间

$$m = \frac{10(t_1 + 2t_2 + 3t_3 + \ldots + 9t_9) - 45(t_1 + t_2 + t_3 + \ldots + t_9)}{825}$$

$$= \frac{2(t_1 + 2t_2 + 3t_3 + \ldots + 9t_9) - 9(t_1 + t_2 + t_3 + \ldots + t_9)}{165}$$

UxBRG 计算

回归线的斜率 *m* 是每一个位周期 *p* 内的定时器计数值。如果定时器设定为对指令周期 TCY（TCY = 1/FCY）进行计数，此时位周期 *p* 表达式如公式 A-3 所示。位周期的倒数即波特率，如公式 A-4 所示。

公式 A-3： 位周期

$$p = m \times Tcy$$

公式 A-4： 测量的波特率

$$baud = \frac{1}{p} = \frac{1}{m \times Tcy} = \frac{Fcy}{m}$$

UART 的波特率由 UxBRG 寄存器设定值进行控制。如《*dsPIC30F 系列参考手册*》中所给出，公式 A-5 给出了基于 UxBRG 的波特率表达式。

公式 A-5： 给定的波特率

$$baud = \frac{Fcy}{16(UxBRG + 1)}$$

结合公式 A-4 和公式 A-5，可以得出基于斜率 *m* 的 UxBRG 值表达式，而该值可根据输入捕捉时间 t_0,t_1, t_2... t_9 来计算。

公式 A-6 可推导为：

公式 A-6： UxBRG 值

$$UxBRG = \frac{m}{16} - 1$$

结合公式 A-2 和公式 A-6 可得出与输入信号波特率相应的UxBRG 寄存器值的简单计算方法，如公式 A-7所示。

公式 A-7： 完整计算

$$UxBRG = \frac{2(t_1 + 2t_2 + 3t_3 + \ldots + 9t_9) - 9(t_1 + t_2 + t_3 + \ldots + t_9)}{2640} - 1$$

简化的 UxBRG 计算

当运算速度至关重要时，位时间亦即信号的斜率 *m* 可如下面公式 A-8 所示，根据两个时间测量值快速计算出来。

公式 A-8：　　位时间的简化计算

$$m = \frac{(t_8 - t_0)}{8}$$

该公式使用两个下降沿的时间以抵消上升时间、下降时间以及传播延迟的差异。通过将 8 个位周期进行平均也可使得除法操作更为方便，这是因为除法可通过简单的右移来实现。结合公式 A-6 和公式 A-8 可得出 UxBRG 的最终简化计算公式，如以下公式 A-9 所示。

公式 A-9：　　UxBRG 的简化计算

$$UxBRG = \frac{(t_8 - t_0)}{128} - 1$$

误差计算

可使用几种不同统计技术来表征测量值偏离其期望值的程度。可以计算相关系数 r 或标准偏差 δ，但上述计算存在计算量大且耗时较长的问题，这是因为需进行几个平方函数以及一个平方根的运算。一个更为简单的方法是采用平均绝对误差，公式 A-10 所示为误差绝对值的平均值。

公式 A-10：　　平均绝对误差

$$MAE = \frac{1}{n}\sum |x - \bar{x}|$$

符号 $\bar{x}$ 是被测信号的期望值。可根据图 A-2 来确定所有时间t_0,t_1, t_2... t_9的期望值。图中的斜率*m*可根据公式 A-2 计算，而 y 轴截距 *b* 可通过公式 A-11 进行计算。

公式 A-11：　　Y 截距

$$b = \frac{\sum y - m\sum x}{n}$$

在该公式中，用 0,1, 2... 9 替换 x ，并用 t_0, t_1, t_2... t_9 替换 y。所有 *x* 值为已知的，且 t_0 = 0 和 n =10，所以公式可简化为公式 A-12。此时，y 轴截距实际为 t_0 的期望值。

公式 A-12：　　Y 截距的期望值

$$\bar{t}_0 = \frac{(t_1 + t_2 + t_3 + \ldots + t_9) - 45m}{10}$$

因为仅需一次乘法和除法操作，而且为获得斜率在公式 A-2 中已经对 (t_1 + t_2 +...+ t_9) 进行了计算，因此上式的计算是非常容易的。

当斜率和截距为已知时，所有期望值都可通过公式 A-13 所示的直线公式进行计算。

公式 A-13：　　期望时间

$$\bar{t}_i = mx_i + \bar{t}_0$$

用 0, 1, 2... 9 替换 x_0, x_1, x_2... x_9 代入上式可以获得所有期望值。由于 x_i 值的递增增量为 1，因此在程序中更容易实现在每一个结果中加入斜率 *m* 以获得下一个值。

附录 B： 使用回归计算的源代码

软件许可协议

Microchip Technology Incorporated （以下简称 “本公司”）在此提供的软件旨在向本公司客户提供专门用于 Microchip 生产的产品的软件。

本软件为本公司及 / 或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。

本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下、因任何原因造成的特殊的、附带的或间接的损害，本公司概不负责。

```
/*****************************************************************
 *
 *               dsPIC30F 自动波特率检测源代码
 *
 *****************************************************************
 * 文件名：          UART Auto Baud by Regression.c
 * 头文件：          p30F6014.h
 *                   math.h
 * 日期：            10/08/2004
 * 处理器：          dsPIC30F6014
 * 编译器：          MPLAB C30 1.20.02
 * 公司：            Microchip Technology, Inc.
 *
 * 软件许可协议
 *
 * Microchip Technology Incorporated （以下简称 “本公司”）在此提供的软件旨在
 * 向本公司客户提供专门用于 Microchip 生产的产品的软件。
 * 本软件为本公司及 / 或其供应商所有，并受到适用的版权法保护。
 * 版权所有。使用时违反前述约束的用户可能会依法受到刑事制裁，
 * 并可能由于违背本许可的条款和条件而承担民事责任。
 *
 *
 *
 *
 *
 *
 * 本软件是按 “现状”提供的。
 * 任何形式的保证，无论是明示的、暗示的或法定的，
 * 包括但不限于有关适销性和特定用途的暗示保证，均不适用于本软件。
 * 对于在任何情况下、因任何原因造成的特殊的、
 * 附带的或间接的损害，本公司概不负责。
 *
 *
 * 作者                  日期         注释
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Mike Garbutt        10/8/2004   原始版本         (Rev 1.0)
 *
 *****************************************************************/

#include "p30F6014.h"              // 标准头文件
#include "math.h"                  // 数学库
#define Fcy  29491200              // 允许波特率计算用于显示

//-----------------------------------------------------------------------------
// 函数原型

void SetupAutoBaud(void);          // 该函数用于设置 UART1、 IC1 和 TMR3
void CalculateBaud(void);          // 该函数用于计算 U1BRG 值
```

```
//------------------------------------------------------------------------------
// 变量

unsigned int ICCount = 0;                       // 对捕捉事件次数进行计数
unsigned int T3Count = 0;                       // 对 Timer 3 中断数进行计数
unsigned int CurrentCapture;                    // 记录 UART 边沿的时间
unsigned int PreviousCapture;                   // 存储上一次边沿时间测量值
unsigned int CaptureDifference;                 // UART 边沿时间差值
long SumXY, SumY;                               // 用于回归计算的立即数
long RegressionData[10];                        // 用于线性回归的数据
unsigned long BaudRate;                         // 计算波特率

//------------------------------------------------------------------------------
// 主程序
// 无限循环检测 UART 输入数据 0x55 的波特率
// 并每次在波特率计算完后输出一条报文。

int main(void)
{
    while(1)                                    // 无限循环
    {
        SetupAutoBaud();                        // 为自动波特率检测设置 UART1、 IC1 及 TMR3
        while(U1BRG == 0) {}                    // 等待自动波特率检测结束
        BaudRate = (Fcy / 16) / (U1BRG + 1);    // 查看当前使用的波特率
        printf("Baud rate: %ld\r", BaudRate);   // 输出带有波特率的文本
    while(U1STAbits.TRMT == 0) {}               // 等待发送结束
    }
}                                               // main() 结束

//------------------------------------------------------------------------------
// 设置外设和中断以进行波特率检测

void SetupAutoBaud(void)
{
    U1BRG = 0;                                  // U1BRG 初始值未知
    U1MODE = 0x8020;                            // 使能 UART 中的自动波特率检测功能
    U1STA = 0x0000;                             // 将 UART 的其他功能设定为缺省状态

    ICCount = 0;                                // 对捕捉事件的次数进行初始化
    IC1CON = 0x0000;                            // 复位输入捕捉 1 模块
    IC1CON = 0x0001;                            // 使能输入捕捉 1 模块
    IFS0bits.IC1IF = 0;                         // 清除捕捉模块 1 中断标志
    IEC0bits.IC1IE = 1;                         // 使能捕捉 1 中断

    T3CON = 0x0000;                             //Timer 3 关闭
    IEC0bits.T3IE = 0;                          // 清除 Timer 3 中断使能
    T3Count = 0;                                // 对 Timer 3 的中断数进行初始化
    PR3 = 0xffff;                               //Timer 3 周期为最大值
    T3CON = 0x8000;                             //Timer 3 启用且设置为 1:1 预分频比以及内部时钟
}

//------------------------------------------------------------------------------
// 计算 U1BRG 波特率发生器的值

void CalculateBaud(void)
{
    int i;                                      // 用于进行误差求和的循环变量
    long Slope;                                 // 回归计算的斜率 （位时间）
    long Yintercept;                            // 第一个边沿的期望的 （计算的）时间
    long PlotLine;                              // 每一个边沿的期望的 （计算的）时间
    long SumOfErrors = 0;                       // 所有误差的和 | 测量值 - 期望值 |

    Slope = (2 * SumXY - 9 * SumY) / 165;       // 计算斜率 = 一个位时间
    Yintercept = (SumY - Slope * 45) / 10;      // 计算第一个边沿的期望时间
```

```
    PlotLine = Yintercept;                          // 对每一个边沿的期望时间进行初始化
    for(i=0; i<10; i++)                             // 循环以对所有绝对误差进行累加
    {
        SumOfErrors += abs(RegressionData[i] - PlotLine);// 计算并加上下一个误差
        PlotLine += Slope;                                  // 计算下一个边沿的期望时间
    }

    if((SumOfErrors * 2) < Slope)                   // 检查是否平均绝对误差 <5%
    {                                               // （10 个误差值累加和的两倍是否 < 一个位时间 ?）
        U1BRG = ((Slope + 8) >> 4) - 1;             // 计算 UxBRG（通过加半位进行舍入）
        U1MODE = 0x8000;                            // 使能 UART 并禁止自动波特率检测
        U1STA = 0x0400;                             // 使能发送
    }
    else
    {
    SetupAutoBaud();                                // 误差太大因此重新开始
    }
}
//-----------------------------------------------------------------------------
// 输入捕捉 1 ISR
// 获得时间测量值并加到回归计算所需的累加和中

void _ISR _IC1Interrupt(void)
{
    IFS0bits.IC1IF = 0;                         // 清除捕捉 1 中断标志
    PreviousCapture = CurrentCapture;           // 存储前一次时间测量值
    CurrentCapture = IC1BUF;                    // 获取新的时间测量值
    T3Count = 0;                                // 复位超时计数器
    if(ICCount == 0)                            // 检查是否为第一个边沿
    {
        IFS0bits.T3IF = 0;                      // 清除 Timer 3 中断标志
        IEC0bits.T3IE = 1;                      // 使能 Timer 3 中断用于超时检查
        RegressionData[0] = 0;                  // 第一个边沿时间的初始值
        SumY = 0;                               // 时间测量值和的初始值
        SumXY = 0;                              // 位数 x 时间和的初始值
    }
    else                                        // 检查是否非第一个边沿
    {
        CaptureDifference = CurrentCapture - PreviousCapture; // 获得时间差
        RegressionData[ICCount] = RegressionData[ICCount-1] + CaptureDifference;
                                                // 将时间差加到前一次时间测量值中
        SumY += RegressionData[ICCount];        // 对时间测量值进行求和
        SumXY += RegressionData[ICCount] * ICCount;// 对位数 x 时间测量值进行求和
    }
    ICCount++;                                  // 对边沿递增计数
    if(ICCount == 10)                           // 检查是否最后一个边沿
    {
        IEC0bits.IC1IE = 0;                     // 清除捕捉 1 中断使能位
        IEC0bits.T3IE = 0;                      // 清除 Timer 3 中断使能位
        CalculateBaud();                        // 计算 U1BRG 值并使能 UART
    }
}
//-----------------------------------------------------------------------------
//Timer 3 ISR
// 检查是否超时，即自前一次输入捕捉是否有两次溢出

void _ISR _T3Interrupt(void)
{
   IFS0bits.T3IF = 0;               // 清除 Timer 3 中断标志
   T3Count++;                       // 自上一次捕捉起的中断递增计数
   if(T3Count == 2)                 // 检查自上一次捕捉起是否发生太多定时器溢出
   {
      SetupAutoBaud();              // 超时因此重新开始
   }
}
```

附录 C： 使用简便计算方法的源代码

```
/*****************************************************************
 *
 *               dsPIC30F 自动波特率源代码
 *
 *****************************************************************
 * 文件名：       UART Auto Baud by Simple Calculation.c
 * 头文件：    p30F6014.h
 *                  math.h
 * 日期：            10/08/2004
 * 处理器：       dsPIC30F6014
 * 编译器：        MPLAB C30 1.20.02
 * 公司：         Microchip Technology, Inc.
 *
 * 软件许可协议
 *
 * Microchip Technology Incorporated （以下简称 “本公司”）在此提供的软件旨在
 * 向本公司客户提供专门用于 Microchip 生产的产品的软件。
 * 本软件为本公司及 / 或其供应商所有，并受到适用的版权法保护。
 * 版权所有。使用时违反前述约束的用户可能会依法受到刑事制裁，
 * 并可能由于违背本许可的条款和条件而承担民事责任。
 *
 *
 *
 *
 *
 *
 * 本软件是按 “现状”提供的。
 * 任何形式的保证，无论是明示的、暗示的或法定的，
 * 包括但不限于有关适销性和特定用途的暗示保证，均不适用于本软件。
 * 对于在任何情况下、因任何原因造成的特殊的、
 * 附带的或间接的损害，本公司概不负责。
 *
 *
 * 作者                日期            注释
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Mike Garbutt        10/8/2004       原始版本         (Rev 1.0)
 *
 *****************************************************************/

#include "p30F6014.h"                   // 标准头文件
#include "math.h"                       // 数学库
#define Fcy  29491200                   // 允许波特率计算用于显示

//---------------------------------------------------------------------------
//Prototypes

void SetupAutoBaud(void);               // 该函数用于设置 UART1、 IC1 和 TMR3
void CalculateBaud(void);               // 该函数用于计算 U1BRG 值

//---------------------------------------------------------------------------
//Variables

unsigned int ICCount = 0;               // 对捕捉事件次数进行计数
unsigned int T3Count = 0;               // 对 Timer 3 中断数进行计数
unsigned int CurrentCapture;            // 记录 UART 边沿的时间
unsigned int PreviousCapture;           // 存储前一次边沿时间测量值
unsigned int CaptureDifference;         //UART 边沿时间的差值
unsigned long SumOfBitTimes;            //
unsigned long BaudRate;                 // 计算波特率
```

```
//------------------------------------------------------------------------------
//Main 程序
/ 无限循环检测 UART 输入数据 0x55 的波特率
// 并每次在波特率计算完后输出一条报文。

int main(void)
{
    while(1)                                    // 无限循环
    {
        SetupAutoBaud();                        // 设置 UART1、 IC1 和 TMR3 用于自动波特率检测
        while(U1BRG == 0) {}                    // 等待自动波特率检测结束
        BaudRate = (Fcy / 16) / (U1BRG + 1);    // 查看当前使用的波特率
        printf("Baud rate: %ld\r", BaudRate); // 输出带波特率的文本
        while(U1STAbits.TRMT == 0) {}           // 等待发送结束
    }
}                                               // main() 结束
//------------------------------------------------------------------------------
// 设置外设以及中断以执行波特率检测

void SetupAutoBaud(void)
{
    U1BRG = 0;                                  //U1BRG 初始值为未知
    U1MODE = 0x8020;                            // 使能 UART 中的自动波特率检测功能
    U1STA = 0x0000;                             // 将 UART 其余功能设定为缺省状态

    ICCount = 0;                                // 初始化捕捉事件的次数
    IC1CON = 0x0000;                            // 复位输入捕捉 1 模块
    IC1CON = 0x0001;                            // 使能输入捕捉 1 模块
    IFS0bits.IC1IF = 0;                         // 清除捕捉 1 中断标志
    IEC0bits.IC1IE = 1;                         // 使能捕捉 1 中断

    T3CON = 0x0000;                             //Timer 3 关闭
    IEC0bits.T3IE = 0;                          // 清除 Timer 3 中断使能位
    T3Count = 0;                                // 初始化 Timer 3 中断数
    PR3 = 0xffff;                               //Timer 3 周期是最大值
    T3CON = 0x8000;                             //Timer 3启用且设置为1:1预分频比以及内部时钟
}
//------------------------------------------------------------------------------
// 输入捕捉 1 ISR
// 获得时间测量值并加入到位时间和中
// 在求出 8 个位时间之和后计算 U1BRG 值

void _ISR _IC1Interrupt(void)
{
    IFS0bits.IC1IF = 0;                         // 清除捕捉 1 中断标志
    PreviousCapture = CurrentCapture;           // 存储上一次时间测量值
    CurrentCapture = IC1BUF;                    // 获得新的时间测量值
    T3Count = 0;                                // 复位超时计数器
    if(ICCount == 0)                            // 检查是否为第一个边沿
    {
        IFS0bits.T3IF = 0;                      // 清除 Timer 3 中断标志
        IEC0bits.T3IE = 1;                      // 使能用于超时检查的 Timer 3 中断
        SumOfBitTimes = 0;                      // 位时间和的初始值
    }
    else                                        // 检查是否非第一个边沿
    {
        if(ICCount != 9)                        // 检查是否非最后一个边沿
        {
            CaptureDifference = CurrentCapture - PreviousCapture; // 获取时间差
            SumOfBitTimes += CaptureDifference; // 将时间差加入时间和
        }
    else                                        // 检查是否为最后一个边沿
        {
            IEC0bits.IC1IE = 0;                 // 清除捕捉 1 中断使能位
            IEC0bits.T3IE = 0;                  // 清除 Timer 3 中断使能位
```

```
            U1BRG = ((SumOfBitTimes + 64) >> 7) - 1; // 计算 UxBRG（带舍入功能）
            U1MODE = 0x8000;                // 使能 UART 并禁止自动波特率检测
            U1STA = 0x0400;                 // 使能发送
        }
    }
    ICCount++;                              // 边沿的递增计数
}
//-----------------------------------------------------------------------------
//Timer 3 ISR
// 检查超时，即自前一次输入捕捉起是否有两次溢出

void _ISR _T3Interrupt(void)
{
    IFS0bits.T3IF = 0;                      // 清除 Timer 3 中断标志
    T3Count++;                              // 自上一次捕捉起递增中断计数
    if(T3Count == 2)                        // 检查自上一次捕捉起是否发生太多定时器溢出
    {
        SetupAutoBaud();                    // 超时，重新开始
    }
}
```

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 及位于加利福尼亚州 Mountain View 的全球总部、设计中心和晶圆生产厂均于 2003 年 10 月通过了 ISO/TS-16949:2002 质量体系认证。公司在 PICmicro® 8 位单片机、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Alpharetta, GA
Tel: 1-770-640-0034
Fax: 1-770-640-0307

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣何塞 San Jose
Mountain View, CA
Tel: 1-650-215-1444
Fax: 1-650-961-0286

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

中国 – 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

印度 India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Penang
Tel:604-646-8870
Fax:604-646-5086

菲律宾 Philippines - Manila
Tel: 011-632-634-9065
Fax: 011-632-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-352-30-52
Fax: 34-91-352-11-47

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

07/01/05