AN2290

8位PIC®单片机的位拆裂增强型UART

作者： *Mary Tamar Tan*
Microchip Technology Inc.

简介

大多数8位PIC®单片机具有一个或多个片上通用异步收发器（Universal Asynchronous Receiver Transmitter，UART）。但是在没有UART硬件可用的情况下，或者需要额外的串行通信接口的情况下，位拆裂将是最佳选择。位拆裂是一种用于通过软件（而非专用硬件外设）创建串行I/O通信接口的技术。数据发送和接收几乎完全由软件控制。其中包括采样、电平检测、定时、同步、缓冲区控制、驱动程序状态切换和错误检测。

本应用笔记重点介绍8位PIC单片机上实现的位拆裂UART驱动程序，还讨论了计算、性能和精度因素、限制以及固件详细信息。应当注意的是，驱动程序中使用了几个硬件外设、Timer0和电平变化中断引脚，以便获得更精确的时序并缩短处理时间。本应用笔记演示了如何使用MPLAB®代码配置器（MPLAB® Code Configurator，MCC）配置这些外设。按照本文档中介绍的详细步骤，用户应该能够在短短几分钟内设置位拆裂UART驱动程序。

该驱动程序还具有增强功能，可完全兼容现有MCC LIN协议栈库。因此，本文档中的驱动程序将称为位拆裂增强型UART（EUART）。若用户对使用LIN库实现驱动程序感兴趣，可参见AN2059《LIN基础知识和8位PIC®单片机上实现的MCC LIN协议栈库》（DS00002059B_CN）来了解详细信息。

特性

位拆裂EUART包含以下功能：

- 半双工异步收发
- 8 MHz时钟频率下最高38.4K波特率
- 使用IOC引脚进行中断驱动接收
- 位周期取决于Timer0中断
- 虚假启动检测
- 输入缓冲区溢出错误检测
- 接收字符帧错误检测
- 自动错误处理机制

位拆裂EUART通过实现以下附加功能来支持MCC LIN驱动程序：

- 自动检测和校准波特率
- LIN从节点的间断检测
- LIN主节点的间断传输

资源使用

使用几种片上外设来保持稳健的通信和精确时序。Timer0用于保持每位发送和接收的时序。定时器预分频器经调整可确保EUART以所需的波特率运行。电平变化中断（Interrupt-on-Change，IOC）引脚用于每个接收字节的起始位（下降沿）检测。IOC也用于检测LIN总线系统中同步字符的五个上升（正）沿。

处理时间

驱动程序用于执行EUART任务的处理时间受比特率、时钟频率和代码设计的影响。与轮询方法相比，使用中断可使EUART耗费的处理时间少很多。对于发送或接收的单个字符，驱动程序实际上仅使用整个字符持续时间内CPU处理时间的一部分。例如，当移出一位时，驱动程序只在每个定时器中断时执行任务（即，将引脚设置为高电平或低电平），并最终释放CPU供其他应用程序使用。非UART任务的可用处理时间（$T_{NON-UART}$）以百分比表示，其计算方式如公式1所示。

公式1：　　可用处理时间百分比

$$可用处理时间百分比 = \left(1 - \frac{T_{UART}}{T_{Char}}\right) \times 100\%$$

其中，

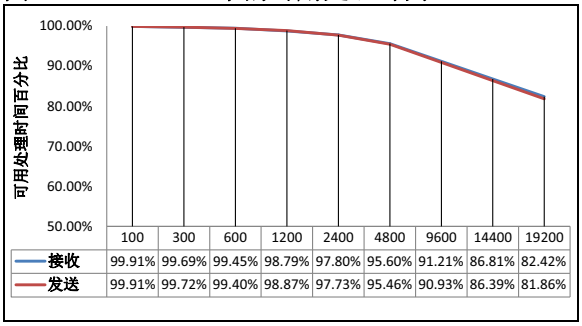
TUART = 仿真每个字符的EUART位信号所需的总时间

TCHAR = TUART + TNON-UART

请注意，由于代码执行的原因，测量的处理时间可能因发送或接收的字符而有所不同。将引脚设置为“高电平”所需的时间与将引脚设置为“低电平”不同，并且采样“高电平”所需的时间与采样“低电平”也不同。换句话说，用户应预计到字符“A”和字符“Z”的处理时间存在细微差异。由于EUART由中断驱动，因此对于特定的波特率，只有TUART和TNON-UART可能变化，而TCHAR始终保持不变。

图1给出了使用16 MHz INTOSC、以4 MHz运行的PIC16F1718器件的近似可用处理时间。测量在发送和接收ASCII字符“M”（4Dh）的过程中进行。

图1：　　4 MHz下的可用处理时间



	100	300	600	1200	2400	4800	9600	14400	19200
接收	99.91%	99.69%	99.45%	98.79%	97.80%	95.60%	91.21%	86.81%	82.42%
发送	99.91%	99.72%	99.40%	98.87%	97.73%	95.46%	90.93%	86.39%	81.86%

从图1可以看出，可用处理时间百分比随着波特率的增加而减少。这是因为对于特定的时钟频率，TUART对于所有波特率是恒定的，而TCHAR随着波特率的增加而减少。

异步发送和接收

位拆裂EUART采用标准非归零（non-return-to-zero，NRZ）格式发送和接收数据。连续发送相同值的数据位时，将保持该位的输出电平不变，而不会在发送完每个位后返回到中间电平。

每个字符发送包含1个启动位及随后的8个数据位，并始终由1个停止位终止。启动位始终处于“空格”（低电平）状态，停止位始终处于“标记”（高电平）状态。发送端口在标记状态空闲。每个发送位持续时间为1/(波特率)。

EUART首先发送和接收LSb。在半双工模式下，无法同时执行发送和接收。它们只能共用相同的数据格式和波特率。

波特率

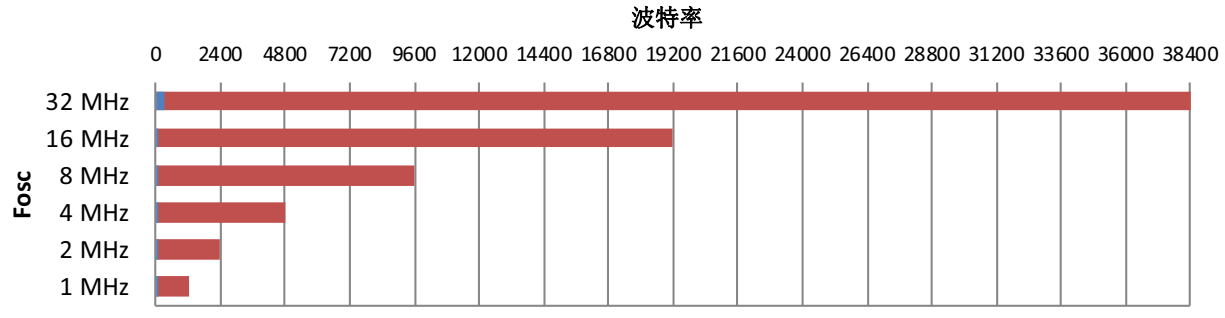
波特率定义通过串行端口传输数据的速率。在EUART中，波特率等同于比特率，这意味着它也是在1秒内移入或移出串行端口的数据位数。

注：　术语“波特率”和“比特率”将在本文档中互换使用。它们在本文中的字面意义相同。

波特率限值

确定位拆裂EUART波特率限值的主要参数之一是时钟频率（FCLK）。对于8位PIC MCU，它等于FOSC/4，其中FOSC是器件振荡器频率。图2给出了使用不同FOSC时建议的波特率限值。

图2：　建议的波特率限值



	1 MHz	2 MHz	4 MHz	8 MHz	16 MHz	32 MHz
最大波特率	1200	2400	4800	9600	19200	38400
最小波特率	100	100	100	100	100	300

通常，建议的最小波特率为100，除非使用32 MHz振荡器，此时指令执行速度太快而使驱动程序无法处理。图2指出了振荡器频率和最大可实现波特率之间的直接关联。可以观察到，随着器件频率加倍，最大波特率也增加大约一倍。可以在推荐限值范围外工作，但不能保证驱动程序的行为。这是因为受到几个硬件和软件因素的限制，这些因素将在“性能和精度”部分中讨论。

附录A：“波特率与实际速率”部分给出了在不同器件频率下运行的PIC16F1718单片机的所需波特率与实际波特率之间的比较示例。

最大和最小波特率

可实现的最大和最小波特率主要取决于所选时钟源的频率。使用Timer0时，最大波特率和最小波特率（Fosc的函数）分别用公式2和公式3表示。

公式2：　给定Fosc的最大波特率

$$Baud_{Max} = \frac{F_{OSC}}{4 \times (256 - 255) \times 预分频比}$$

$$Baud_{Max} = \frac{F_{OSC}}{4 \times 预分频比}$$

公式3：　给定Fosc的最小波特率

$$Baud_{Min} = \frac{F_{OSC}}{4 \times 256 \times 预分频比}$$

Timer0重载值

波特率从根本上说是定时器周期的倒数。Timer0实现为8位定时器，在TMR0寄存器每次溢出时触发中断事件。TMR0使用特定值重载，以便在每个位周期内创建一个中断。公式4给出了计算一个位周期的TMR0重载值的方法。

公式4：　一个位周期的TMR0重载值

$$TMR0_{OneBit} = 256 - \frac{F_{OSC}}{4 \times 波特率 \times 预分频比}$$

位拆裂EUART接收器设计为通过起始位启动采样程序。因此，需要确定起始位边沿与起始位中间位置之间的时长。半个位周期的重载值可通过公式5获得。

公式5：　半个位周期的TMR0重载值的计算方法

$$TMR0_{HalfBit} = TMR0_{OneBit} + \frac{256 - TMR0_{OneBit}}{2}$$

$$TMR0_{HalfBit} = 128 + \frac{TMR0_{OneBit}}{2}$$

$$\therefore 128 < TMR0_{HalfBit} < 256$$

公式4和公式5的结果是理想值，因此需要考虑其他参数来加以调整。这包括硬件引起的延时（例如中断延时，写入操作后的两条指令周期延时）和时钟精度。此外，还应考虑固件设计引入的延时。有关计算需要考虑的其他因素，请参见“性能和精度”部分。

Timer0 重载调整

在TMR0实际递增之前将执行几个指令周期。公式6给出了通过校准TMR0重载值来补偿硬件和软件引入的延时的方法。

公式6：　　新的TIMER0重载值

$$TMR0_{New} = TMR0_{Ideal} + \frac{N}{预分频系数}$$

其中：

N = 指令周期数

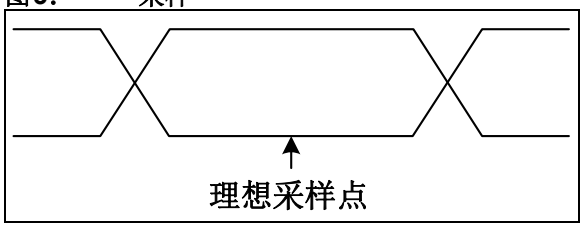
预分频系数 = PS <2:0> + 1

在调整期间，建议仅修改“N”参数。具有较高时序精度的逻辑分析器可以连接到TX引脚，同时发送无效数据来确定发送器是否已经满足所需的波特率。对于接收器，测试引脚可以在进入到退出Timer0中断服务程序（Interrupt Service Routine，ISR）的这段时间内翻转，同时将双通道逻辑分析器连接到测试引脚和RX引脚，以确保采样率处于可接受的公差范围内。

采样

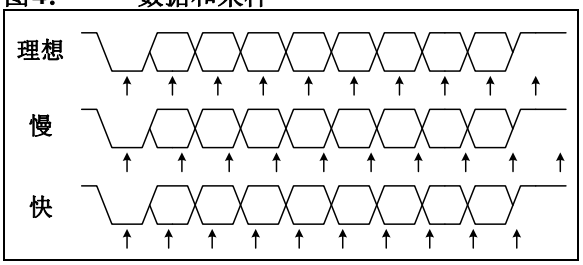
假设理想采样点是传入位的中心，如图3所示。

图3：　　采样



位拆裂EUART实现了8位字符格式，这意味着总共需要对10位（包括起始位和停止位）进行采样。每个位占总时间的10%。因此，如果传入数据速率和接收器采样速率之间的差值超过10%，则在采样过程中会漏掉1位。这将导致接收到的数据不准确。除非出现帧错误，否则驱动程序将始终检测不出这种情况。图4以图形方式给出了由于数据和采样速率不一致而错误解析数据的过程。

图4：　　数据和采样



接收的准确度取决于发送方和接收方的准确度。如果发送方比理想波特率快5%，接收方比理想波特率慢5%，则会出现10%的差值，这必定会漏掉1位。因此，建议每个器件与理想波特率的偏差都不要超过2.5%。

注： 当此位拆裂EUART驱动程序与LIN驱动程序一起使用时，位容差更严格。有关更多详细信息，请参见LIN规范2.2A的“位速率容差”。

性能和精度

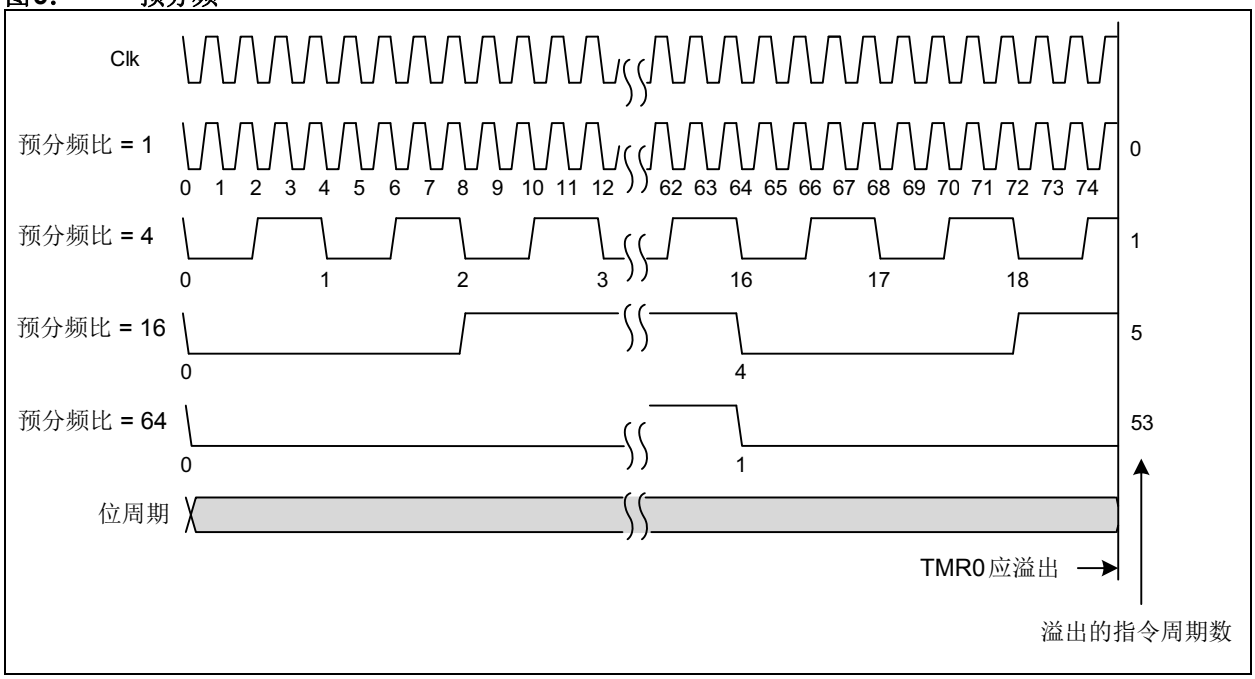
为获得最佳结果，用户应能够考虑可能影响驱动程序的性能和精度的软件和硬件因素。本部分将会介绍其中的部分因素。

预分频

预分频器以特定的固定值将器件时钟频率分频，然后再将其送入定时器模块。Timer0具有独特的8位预分频器，可以配置为从2到256的时钟比。对于特定的器件频率，预分频允许位拆裂EUART以低得多的波特率运行。但是，当预分频比增大时，对于特定波特率，可能的误差量也可能增大。图5给出了在没有预分频器的情况下，TMR0在预期时间内溢出的过程。但是，预分频比为4、16和64时，在定时器溢出之前要执行更多指令周期。必须谨慎选择预分频比，以便实现所需波特率和可能的最佳定时器分辨率。

换句话说，最好使输入时钟源快于所需EUART波特率，然后使用预分频器将速度分频为更慢的所需EUART速度。这将允许在EUART位生成操作之间执行更多指令。

图5：　　预分频



整数算术截断

手动计算的结果与单片机执行的整数算术的结果并不始终完全一致。例如，使用8 MHz振荡器时需要的波特率为9600。将这些值插入公式4将导致TMR0重载值大约等于47.67。可用于TMR0的整数值只有47和48。这将分别导致1.4%和0.7%的误差。

中断响应延时

此位拆裂EUART主要由状态机和中断构建。中断响应延时定义为从发生中断事件到开始执行中断向量处的代码所经过的时间。同步（即定时器）中断的典型响应延时为3至4个指令周期。对于异步中断（即电平变化中断），响应延时为3至5个指令周期，具体取决于中断发生的时间。有关中断响应延时的更多详细信息，请参见具体器件的数据手册。

时钟精度

Timer0在每个指令周期递增，使其高度依赖于所选振荡器的精度。振荡器的频率可能由于V_{DD}以及温度、湿度、压力和老化等环境参数的变化而漂移。PIC单片机的内部振荡器模块经过出厂校准，但频率易受这些因素的影响而发生变化。因此，用户需要根据参考时钟校准器件时钟，以实现极小的时序误差。或者，使用在整个温度和V_{DD}工作电压范围内偏差较小的外部时钟源。

MPLAB® X上的设置

系统要求

- MPLAB® X IDE v3.40或以上版本
- MPLAB®代码配置器（MCC）v3.26或以上版本
- 基础服务库v0.1.15或以上版本

MCC配置

假设用户具有在MPLAB X上创建新项目的基本知识。确保将项目设置为主项目。位拆裂EUART要求使用MCC完成所有硬件和软件初始化。以下步骤举例说明了如何为PIC16F1718配置9600的波特率和16 MHz内部振荡器。

1. 导航到*Tools > Embedded > MPLAB Code Configurator v3*（工具 > 已安装工具 > MPLAB代码配置器v3）以启动MCC版本3。
2. 在Project Resources（项目资源）下，单击**System**（系统），然后选择**System Module**（系统模块）。
3. 选择所需的系统时钟设置。对于示例应用程序，在Oscillator Select（振荡器选择）菜单上选择INTOSC Oscillator: I/O function on CLKIN pin（INTOSC振荡器：CLKIN引脚上的I/O功能）。将FOSC设置为“System Clock”（系统时钟），将16 MHz_HF设置为“Internal Clock”（内部时钟）。取消选中PLL Enabled（使能PLL）、Software PLL Enabled（使能软件PLL）和Low-voltage Programming Enable（使能低电压编程）复选框。
4. 禁止看门狗定时器。
5. 在Device Resources（设备资源）下，选择*Libraries > Foundation Services > SWUART*（库 > 基础服务 > SWUART）。
6. 选择TMR0作为定时器。将波特率设置为9600，发送缓冲区大小设置为8，接收缓冲区大小设置为8。用户应注意实际波特率等于0.0 b/s。要设置实际波特率，TMR0的所需定时器周期应配置为104.17 µs。
7. 选择SWUART时，TMR0模块将自动添加到**Peripherals**（外设）下的**Project Resources**（项目资源）中。
8. 将时钟源设置为FOSC/4。要实现所需的定时器周期，请选中**Enable Prescaler**（使能预分频比）并设置为1:2。将请求的周期设置为104.17 µs。用户将发现实际周期为104 µs。由于“整数算术截断”部分讨论的算术限制，这是最接近周期的可能值。
9. 选中**Enable Timer Interrupt**（允许定时器中断）。
10. 返回*Project Resources > Libraries > Foundation Services > SWUART*（项目资源 > 库 > 基础服务 > SWUART）。实际波特率已更改为9615.38 b/s。

> **注：** 显示的“实际波特率”是基于Timer0的“实际周期”单独计算的，未考虑其他软件和硬件限制。

11. 在本例中，RC6和RC7分别用作TX和RX引脚。要配置这些引脚，请转到**Project Resources**，然后选择**Pin Module**（引脚模块）。
12. 在Pin Manager: Grid [MCC]（引脚管理器：网格[MCC]）下，将**Package**（封装）设置为PDIP28。将RC7设置为输入，将RC6设置为SWUART模块的输出。返回到Pin Module GUI。
13. 要配置TX引脚，确保RC6已设置为输出。禁止Analog（模拟）和WPU，并将IOC设置为“None”（无）。
14. 对于RX引脚，取消选中**Start High**（以高电平启动），**Analog**，**Output**（输出）和**WPU**复选框，并将RC7 IOC设置为“Negative”（负）。这将在该引脚的负边沿上允许IOC。
15. 导航到*Project Resources > System > Interrupt Module*（项目资源 > 系统 > 中断模块）。在GUI中，确保选中“Preemptive interrupt routine”（抢占式中断程序）。TMR0和引脚模块应分别处于第一和第二中断优先级。
16. 在Project Resources菜单附近，单击**Generate**（生成）。首次生成MCC代码时，将出现一个弹出窗口，要求以.mc3文件格式保存配置设置。输入所需的文件名，然后单击**Save**（保存）。
17. MCC配置现已完成。图6汇总了这些设置。

图6： MCC配置设置

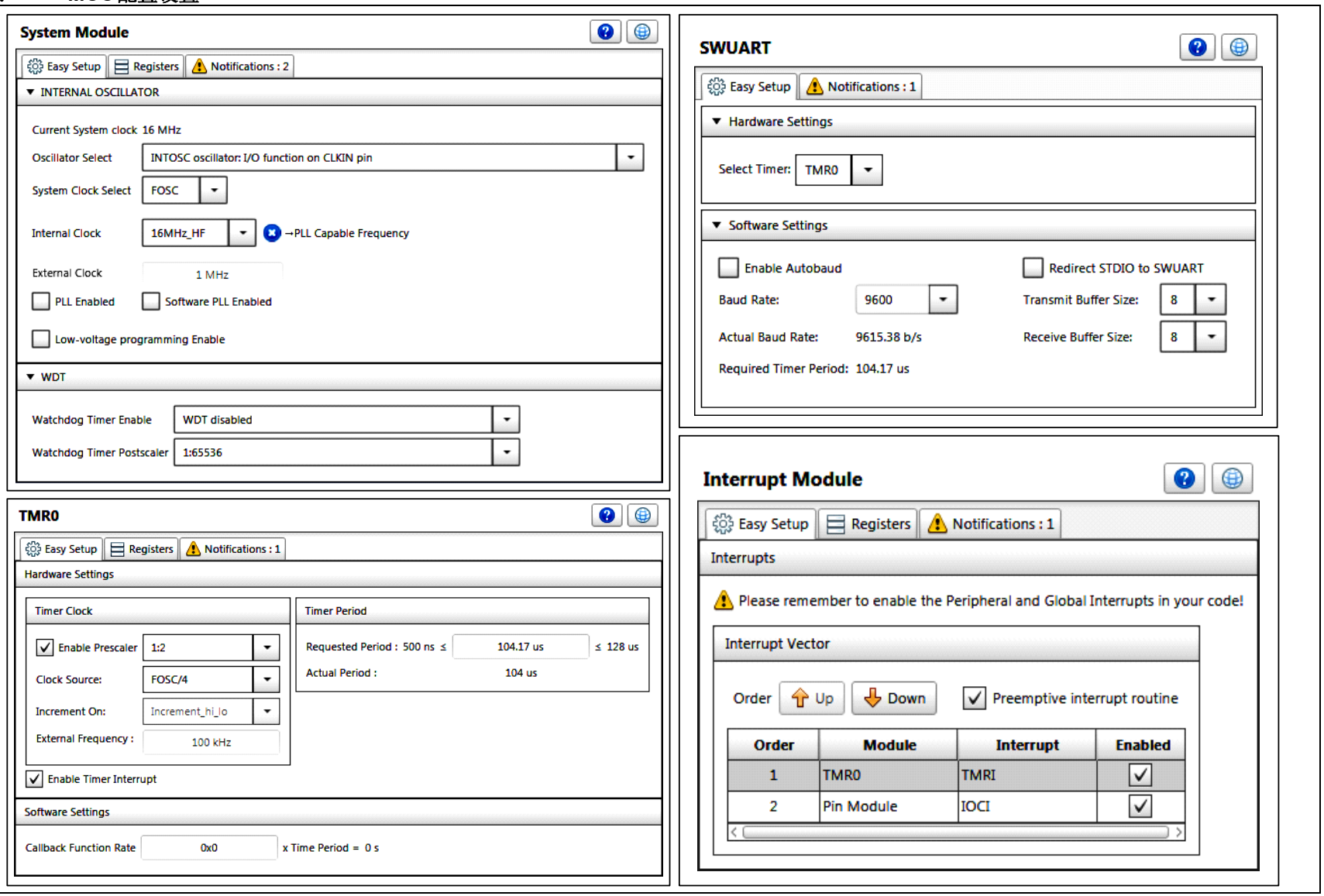
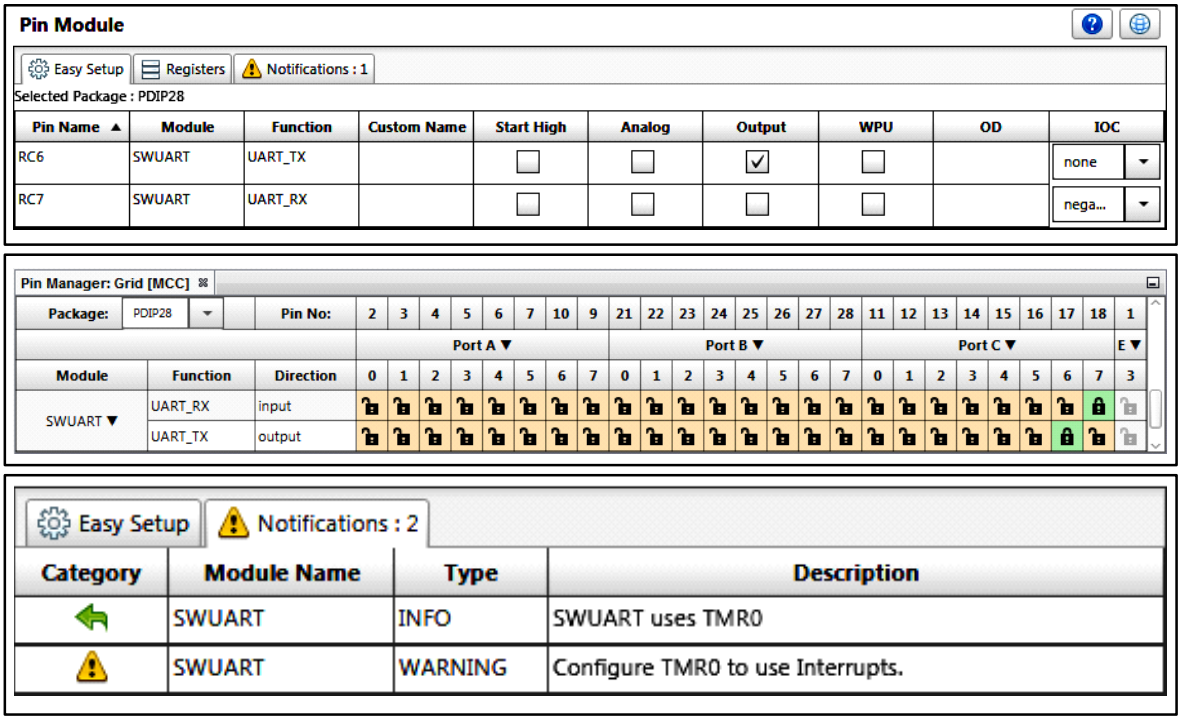


图7：　　MCC引脚设置和SWUART通知

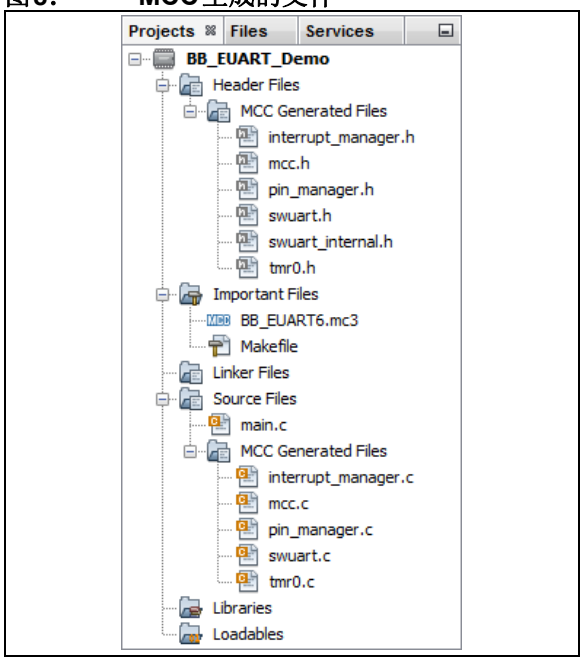


注：　　每个发行版本的MCC GUI外观可能有所不同。以上图片显示的是MCC v3.26和基础服务v0.1.15。

MCC生成的文件

图8给出了在MCC中生成后将自动添加到主项目的所有文件。

图8： MCC生成的文件



将为位拆裂EUART生成2个头文件和1个C文件。

- `swuart.h` ——此文件包含UART的基本功能（如初始化、读取和写入）的所有变量声明和函数原型。
- `swuart_internal.h` ——包含在半双工异步模式下模拟硬件EUSART的行为所需的所有宏。
- `swuart.c` ——包含支持位拆裂EUART的不同基本和增强功能的变量声明和函数。

`swuart.c`中的2个重要值可能需要由用户调整以补偿硬件和软件引入的延时。

- `ONE_BIT_DELAY_COMP` ——补偿1位传输和1位采样间隔期间的延时。尤其是在使用振荡器而不是INTOSC时，可能需要进行修改。
- `HALF_BIT_DELAY_COMP` ——补偿起始位采样期间的延时。尤其是在使用读功能时，可能需要一些调整。只写时，该值并不重要。

初始化期间，将根据设置的补偿器值自动计算新的Timer0重载值（见“Timer0重载调整”部分）。在未来版本中，MCC SWUART驱动程序将支持自动校准。

Main文件

对于新项目，您将注意到MCC会自动生成`main.c`文件。必须执行以下最后步骤才能使驱动程序正确初始化：

1. 打开MCC生成的main.c文件。
2. 取消注释`INTERRUPT_PeripheralInterruptEnable();`和`INTERRUPT_GlobalInterruptEnable();`代码行。
3. 现在即可使用驱动程序。您现在可以在`while(1)`循环中添加应用程序代码。

现在用户已经知道如何使用MPLAB X和MCC设置驱动程序，他/她可能会发现随后的讨论有助于更好地了解驱动程序的工作原理。接下来的部分将介绍固件设计以及有关驱动程序的所有基本详细信息。

固件

位拆裂EUART几乎全部通过软件构建，仅使用少量硬件资源。本部分将概述如何利用这些资源来创建一个完整的独立驱动程序。

发送

图9给出了一个简单的流程图，用于说明在有效写操作后位拆裂EUART如何通过TX引脚异步传输数据。

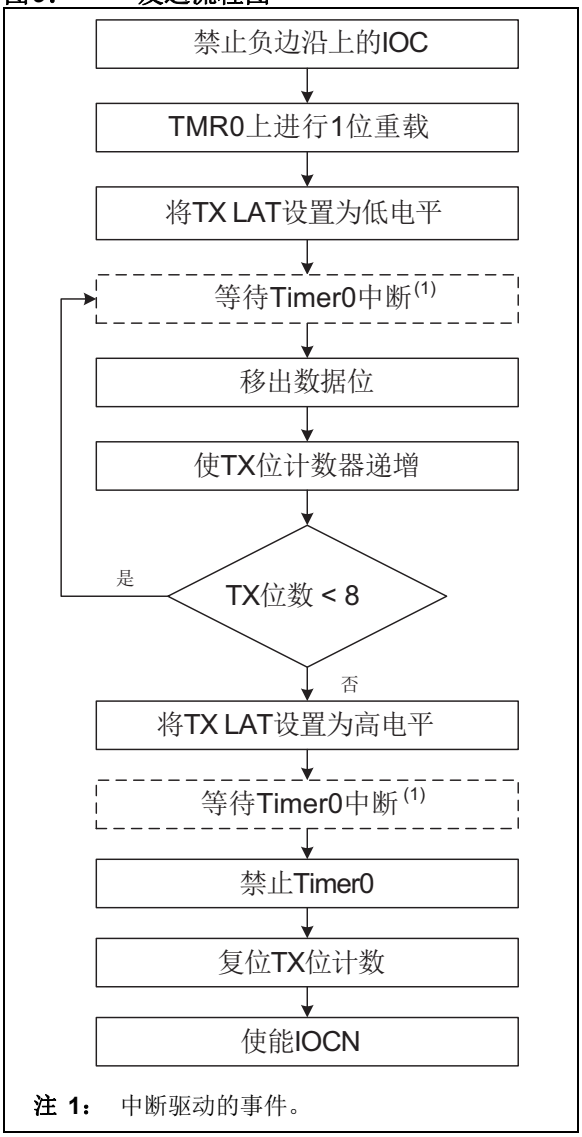
在半双工概念中，通信是双向的，但是不允许两个器件同时传输。因此，在位拆裂EUART中传输之前的第一步是禁止RX引脚上的IOC。当TMR0重载为恰好在1个位周期后中断并且TX引脚拉低以传输起始位时，异步传输开始。驱动程序由中断驱动，允许处理器在等待中断时执行其他重要任务。当Timer0溢出时，会发生中断。每次Timer0中断时，会从TX引脚一次移出1个字符位（从LSb到MSb）。该引脚在第8位传输完成后最终设置为高电平，并在停止位传输的另一个位周期内保持高电平。在完整的数据传输后，Timer0将禁止，IOCN将使能为允许接收。

接收

图10给出了如何通过RX引脚异步接收数据。

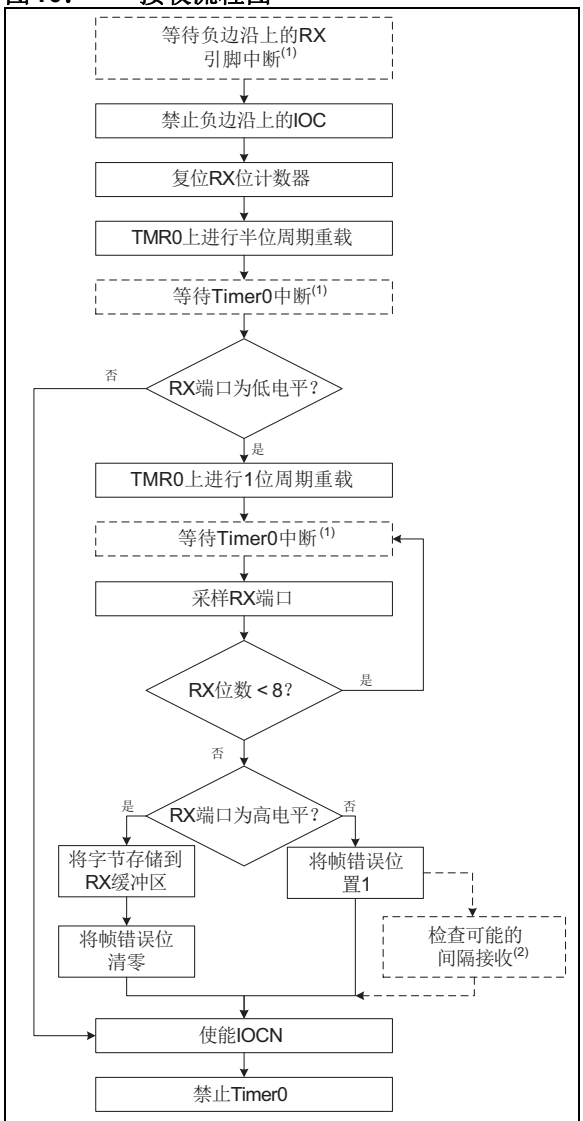
当在RX引脚的负边沿上检测到IOC时，异步接收开始。这意味着在RX引脚上检测到高电平到低电平的跳变，随后发出起始位边沿信号。IOCN随即禁止，TMR0重载为在半个位周期后中断。在第一个Timer0中断后，如果检测到低电平，将对RX引脚进行采样和测试。如果满足上述步骤，则TMR0将重载为在每个位周期中断，以对传入的8个数据位和停止位进行采样。如果接收到停止位，则接收的数据存储在RX FIFO缓冲区中，并且帧错误位被清零。否则，帧错误位将置1。IOCN随后使能并且Timer0禁止以允许另一次接收。

图9：　发送流程图



注 1： 中断驱动的事件。

图10：　接收流程图



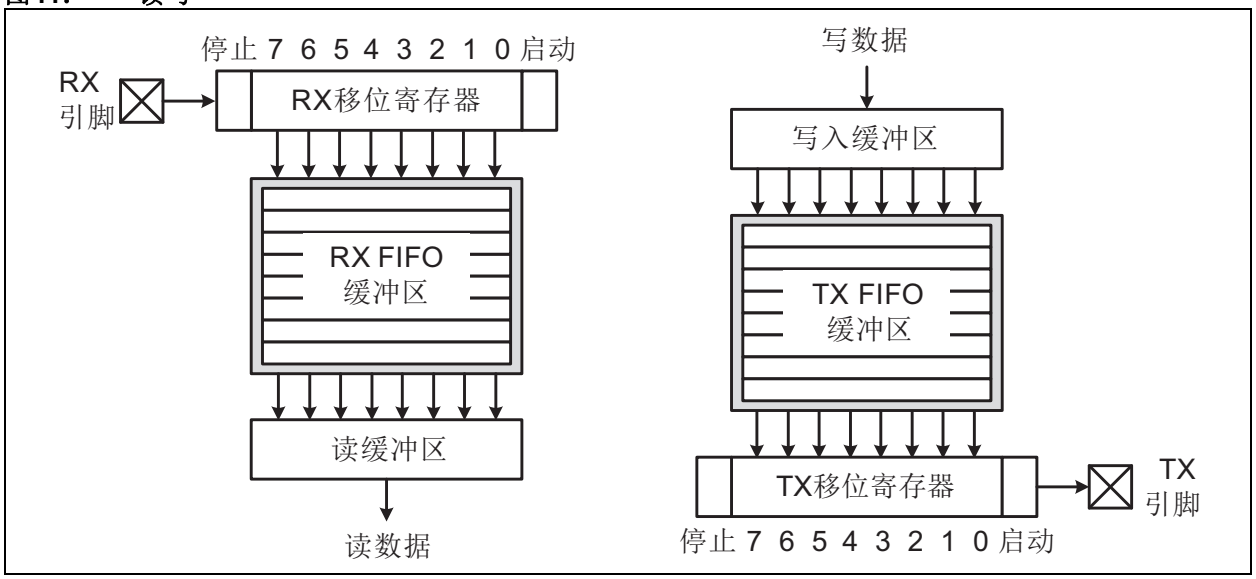
注 1： 中断驱动的事件。

2： 仅适用于LIN从节点。

数据读写

图11显示了读取（左）前和写入（右）后，固件如何处理数据。

图11： 读写



仅当读缓冲区不为空时才会发生有效的读取操作。数据位通过RX引脚移入RX移位寄存器。检测到停止位后，数据将自动传输到RX FIFO缓冲区。RX FIFO存储所有未读数据。在读取期间，FIFO中的顶部未读字符将立即传输到读缓冲区，数据现在可用于其他应用程序。当RX FIFO缓冲区加载未读字符时，位拆裂EUART固件将忽略所有后续传入数据。

数据只能直接写入写缓冲区。当TX FIFO缓冲区和TX移位寄存器均为空时，数据将从写缓冲区自动刷新到TX移位寄存器。如果不为空，则所有写入的数据都在TX FIFO缓冲区中排队。发送完成后，顶部未发送数据将立即传送到TX移位寄存器。如果TX FIFO缓冲区装入未发送字符时发生写操作，数据将停留在写缓冲区，直到停止位从TX引脚移出。

除了缓冲区控制之外，还在固件中实现了错误处理机制。

错误处理

接收期间可能会出现三种错误：虚假启动、帧错误和溢出错误。

虚假启动

当在RX引脚的下降沿上检测到IOC但未接收到起始位时，会发生虚假启动。每当检测到虚假启动时，串行驱动程序状态均设置为空闲，固件则准备好接收另一个字符。

帧错误

在预期时间内未接收到停止位时，会发生帧错误。检测到帧错误后，SW_FERR将置1。固件随后将检查可能的间隔字符接收。SW_FERR在下次读取时清零。

溢出错误

如果RX FIFO缓冲区在加载未读字符时被写入，会发生溢出错误。SW_OERR在缓冲区已满而无法再接收时置1，在读取顶部未读字符后由固件自动清零。

驱动程序详细信息

本部分介绍了驱动程序中实现的不同软件函数、标志位、FIFO缓冲区、移位寄存器和驱动程序状态。

函数

位拆裂EUART由几个固件函数组成，这些固件函数可以分为以下几类：

- 初始化和控制——包括配置、读取、写入、控制和错误处理函数。
- 状态检查 —— 调用这些函数来确定位拆裂 EUART 发送器和接收器的状态。
- 电平变化中断处理——处理中断事件以实现RX引脚上的正、负边沿检测。
- 定时——涉及重载值的计算、调整和设置、Timer0的使能/禁止以及Timer0的中断处理。
- MCC LIN支持——其他函数，支持自动波特率检测、LIN从节点的间隔接收和LIN主节点的间隔发送。这些函数可以使用MCC LIN库而不是硬件EUSART模块来实现位拆裂EUART。

表1： 函数名称

函数名称	参数	返回值	说明
初始化和控制函数			
SWUART_Initialize()	—	—	执行驱动程序的初始化。
SWUART_Read()	—	readValue	返回RX FIFO缓冲区的顶部未读字符。
SWUART_Write()	txdata	-	将数据写入TX FIFO缓冲区。
SWUART_EnableRx()	—	—	禁止Timer0中断并允许RX引脚负边沿上的电平变化中断。
SWUART_CheckRx()	—	—	检查接收数据的帧错误和/或LIN从节点的间隔接收。
SWUART_RxOverrunControl()	—	—	检测RX FIFO缓冲区是否发生溢出。
状态检查			
SWUART_is_tx_ready	—	bool	如果发送FIFO中仍然有空间，则返回真。否则，返回假。
SWUART_is_rx_ready	—	bool	如果RX FIFO不为空，则返回真。否则，返回假。
SWUART_is_tx_done	—	bool	如果所有字符都移出TX移位寄存器，则返回真。否则，返回假。
电平变化中断处理函数			
SWUART_SetIOCNHandler()	SWUART_IOCNHandler	—	将SWUART_IOCNHandler()作为中断服务程序（ISR）的一部分分配给所选RX引脚负边沿上的每个中断。
SWUART_IOCNHandler()	—	—	识别当使能自动波特率时，中断用于开始接收新字节还是同步字符。
SWUART_SetIOCPHandler()	SWUART_IOCPHandler	—	将SWUART_IOCPHandler()作为中断服务程序（ISR）的一部分分配给所选RX引脚正边沿上的每个中断。
SWUART_IOCPHandler()	—	—	正边沿出现ISR时在IOC中调用。用于同步字符接收和自动波特率检测。
定时函数			
SWUART_SetTimerInterruptHandler()	—	—	将SWUART_TimerInterruptHandler()作为中断服务程序（ISR）的一部分分配给Timer0中断。
SWUART_TimerInterruptHandler()	—	—	为每个Timer0中断事件处理代码。
SWUART_SetTimerReload()	reload	—	向TMR0写入一个新的重载值，并允许Timer0中断。
SWUART_CalcOneBitReload()	—	oneBit	基于设置的1位延时补偿器值ONE_BIT_DELAY_COMP返回调整后的1位重载值。
SWUART_CalcHalfBitReload()	—	halfBit	基于设置的半位延时补偿器值HALF_BIT_DELAY_COMP返回调整后的半位重载值。
SWUART_SetTimerPrescaler()(1)	prescaleEnable, prescaleVal	-	使能并设置，或禁止Timer0预分频器。
MCC LIN支持函数			
SWUART_SetAutoBaud()(1)	—	—	适用于LIN从节点。同步接收期间，在RX引脚的第5个上升沿之后捕捉TMR0值。控制自动波特率检测。
SWUART_CalcBaudRate()(1)	periodCapture	—	适用于LIN从节点。基于捕捉的同步字符周期识别新的1位和半位重载值。
SWUART_SendBreak()	—	—	适用于LIN主节点。在无效写入之后启动间隔字符的传输。

注 1： 仅在使能自动波特率时生成。

标志位

表2总结了位拆裂EUART中使用的串行标志位。这些标志位与硬件EUSART对应部分执行相似的功能。

表2： 串行标志位

标志位	说明	功能	硬件等效位
SW_TRMT	发送移位寄存器状态位	指示发送移位寄存器的状态。当发送移位寄存器为空并且TX FIFO中没有待处理字符时，由固件自动置1。当字符从写缓冲区刷新到发送移位寄存器时清零。	TRMT
SW_OERR	溢出错误位	发生RX FIFO缓冲区溢出时置1。在读取顶部字符后，由固件自动清零。	OERR
SW_FERR	帧错误位	在接收期间检测到帧错误时置1。由固件自动清零和置1。	FERR
SW_SENDB	发送间隔字符位	将该位置1意味着在下次传输时将发送间隔字符。完成后由固件自动清零。	SENDB
SW_ABDEN	自动波特率检测使能位	将该位置1会使能自动波特率检测。自动波特率完成时自动清零。	ABDEN

FIFO缓冲区

位拆裂EUART使用2个先进先出（First-in-First-out，FIFO）缓冲区：发送和接收。两个缓冲区的默认大小为8，但可以通过软件或MCC SWUART GUI进行修改。

表3： 串行FIFO缓冲区

FIFO缓冲区	说明	功能	硬件等效位
swuartTxBuffer[]	发送FIFO	发送过程中字符写入（排队）的位置。	TXREG
swuartRxBuffer[]	接收FIFO	读取前字符存储（排队）的位置。	RCREG

移位寄存器

固件上实现2个移位寄存器，分别用于发送和接收。

表4： 串行移位寄存器

移位寄存器	说明	功能	硬件等效位
swuartTxData	发送移位寄存器	将数据位从TX引脚移出（从LSb到MSb）。	TSR
swuartRxData	接收移位寄存器	来自RX引脚的传入位移入的位置。	RSR

驱动程序状态

每个Timer0中断的处理方式是在发送、接收或空闲条件下进入不同的可能状态之一。表5中总结了驱动程序的状态及其定义。图12和图13分别给出了发送和接收状态图。

表5： 串行驱动程序状态

串行状态	说明
发送	
SERIAL_SEND_START_BIT	在对写缓冲区进行有效写入后进入此状态。将线路拉“低”1个位周期。
SERIAL_SEND_BYTE	包含发送移位寄存器。所有8位移出后，将TX引脚拉“高”。
SERIAL_SEND_STOP_BIT	如果TX FIFO缓冲区不为空，则准备另一次发送。否则，通过允许RX引脚负边沿上的IOC来使能接收，并将SW_TRMT标志位置1来使能另一次发送。
SERIAL_SEND_LIN_BRK	在SW_SENDB置1且向TX缓冲区写入一个无效字节后进入此状态。使TX引脚保持“低电平”并持续13个位周期。
接收	
SERIAL_RCV_START_BIT	在RX引脚的负边沿上出现IOC的半个位周期后进入。验证是否接收到起始位。
SERIAL_RCV_BYTE	包含接收移位寄存器。在每个1位定时器中断时，采样来自RX引脚的传入位。所有8位移入后，将立即传送到RX FIFO缓冲区。
SERIAL_RCV_STOP_BIT	验证是否接收到停止位。检查帧错误和缓冲区溢出错误。
SERIAL_RCV_LIN_BRK	当接收到00h并检测到帧错误（SW_FERR = 1）时进入此状态。
SERIAL_RCV_LIN_SYNC	计数自动波特率检测期间Timer0溢出的次数。仅在使能自动波特率（SW_ABDEN = 1）时进入此状态。
空闲	
SERIAL_IDLE	默认状态。未进行发送和接收时进入此状态。

图12： 发送状态图

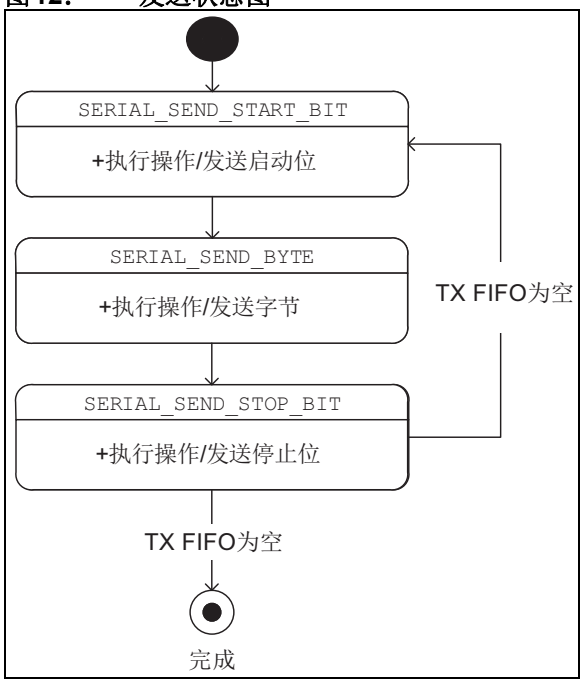
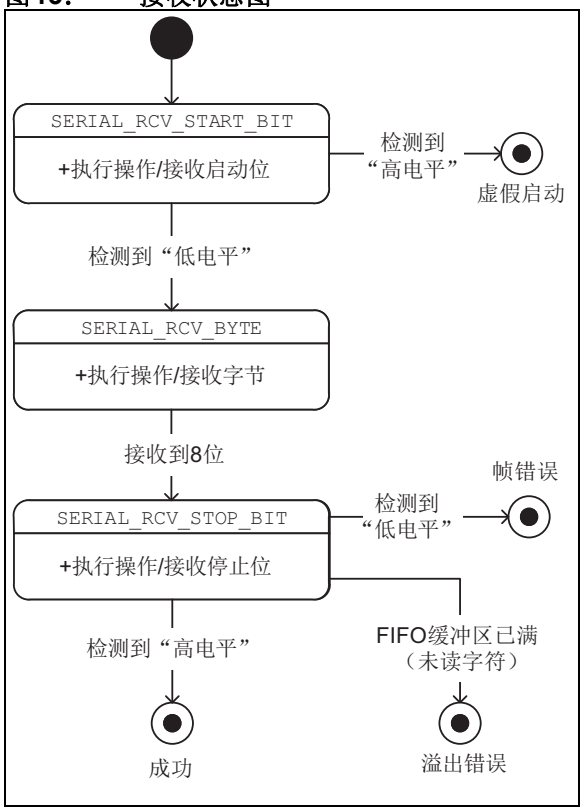


图13： 接收状态图



增强型功能

本部分介绍了用于支持MCC LIN库的位拆裂EUART的附加功能。

间隔字符发送

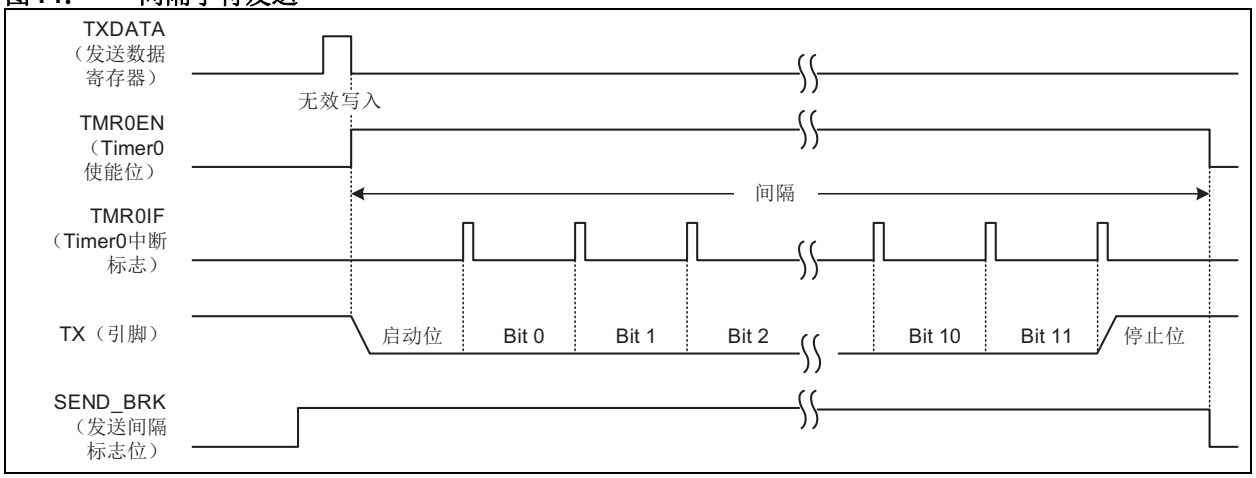
位拆裂EUART可以发送符合LIN总线标准的特殊间隔字符序列（见图14）。间隔字符包括一个起始位以及随后的12个0位和一个停止位。EUART将起始位作为所需的13位时长标称信号的第1位，将停止位作为间隔定界符（至少1位时长的隐性信号）。

要发送间隔字符，将SW_SENDB串行标志位置1。写入发送数据寄存器的值会被忽略，而会发送全0。在发送了相应的停止位之后，固件会自动将SW_SENDB串行标志位复位。

当使用MCC LIN主驱动程序实现时，位拆裂EUART会按照下列顺序来进行间隔发送：

1. 将串行标志SW_SENDB置1以使能间隔序列。
2. 使用无效字符（该值被忽略）装入发送数据寄存器，以启动发送。
3. 装入TMR0并使能TMR0IE，以在一个位周期后中断。
4. 每个Timer0中断事件时，都会自动清零TMRIF。
5. TX引脚设置为低电平并持续至第13个中断，然后在第13个中断时最终设置为高电平。
6. 在第14个中断时，Timer0中断禁止，SW_SENDB自动清零。

图14：　间隔字符发送



间隔字符接收

中断字符的接收遵循与图10中相同的流程图，但插入了一些其他任务。固件检查接收到的字符是否等于00h。如果相等，则传入位的采样将持续至接收到第13个空格。假设TMR0已初始化为在预期的位周期内中断。

发生以下情况时，表示接收到间隔字符：

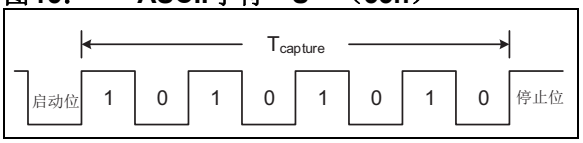
- 允许Timer0中断（TMR0IE = 1）
- 帧错误串行标志位置1（SW_FERR = 1）
- RX移位寄存器清零（swuartRxData = 00h）

使用自动波特率时，建议将Timer0周期设置为与预期位周期的偏差 < ±14%。

自动波特率检测

位拆裂EUART支持波特率自动检测和校准。此功能旨在支持LIN从节点的自动波特率要求。Timer0模块用于捕捉值为55h（ASCII“U”）的同步字符的周期。该字符的独特之处在于它有5个上升沿（包括停止位边沿），如图15所示。

图15：　ASCII字符“U”（55h）



使用电平变化中断（IOC）引脚检测上升沿。在检测到第1个上升沿不久后，TMR0设置为00h。在第5个上升沿，捕捉定时器值。定时器周期取决于捕捉到的值、预分频比和定时器溢出次数，如公式7所示。

公式7：　　使用TIMER0捕捉的周期

$$T_{Capture} = \frac{4 \times (TMR0 + (溢出次数 \times 256)) \times 预分频比}{F_{OSC}}$$

要计算实际位采样时间，捕捉的周期必须除以8位。

公式8：　　1个位周期

$$T_{Bit} = \frac{T_{Capture}}{8}$$

由于波特率等于1/TBIT，结合公式7和公式8将推导出一个表示检测到的波特率与上述其他参数之间关系的公式（公式9）。

公式9：　　检测到的波特率

$$Baud_{Detected} = \frac{F_{OSC} \times 8}{4 \times (TMR0 + (溢出次数 \times 256)) \times 预分频比}$$

为方便起见，在同步接收期间，位拆裂EUART使用默认预分频值8。得到的捕捉值将自动对应于没有预分频比的1位周期。检测到的波特率的公式可以简化，如公式10所示。

公式10：　　预分频比 = 8时检测到的波特率

$$Baud_{Detected} = \frac{F_{OSC}}{4 \times (TMR0 + (溢出次数 \times 256))}$$

随后自动计算1位和半位采样的重载和预分频值。位拆裂EUART仅在第一次间隔-同步接收时校准LIN从机波特率。相对检测误差可使用公式11计算。

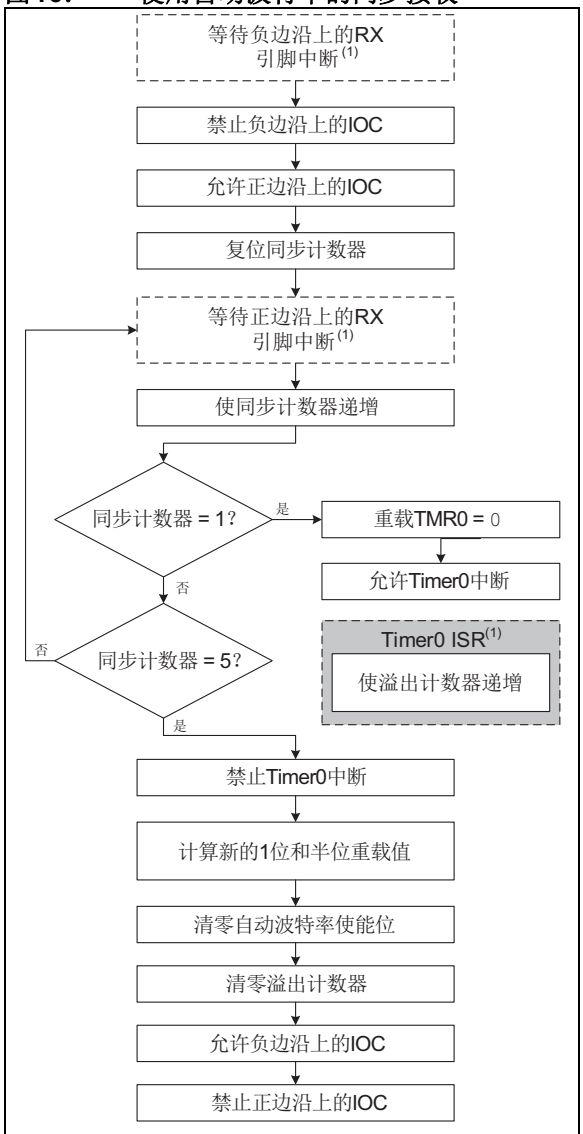
公式11：　　相对波特率检测误差

$$误差百分比 = \frac{Baud_{Detected} - Baud_{Input}}{Baud_{Input}} \times 100\%$$

$Baud_{INPUT}$是传入波特率（即LIN总线系统的主节点波特率）。根据LIN规范，同步后从节点比特率容差相对于主节点比特率的偏差必须< ±2%。

公式10仅提供理想情况下检测到的波特率。有关通过Timer0校准波特率的信息，请参见“Timer0重载调整”部分。图16给出了使能自动波特率时同步字符接收的简化流程图。

图16：　　使用自动波特率的同步接收



注 1：　中断驱动的事件。

结论

本应用笔记介绍了有关如何使用MPLAB X IDE、MPLAB代码配置器（MCC）和8位PIC单片机实现位拆裂EUART的所有必要信息。用户必须熟悉驱动程序的特性，以最大限度发挥其功能并了解其限制。对于可能需要修改固件以在各种应用中进行校准和使用的用户，还提供了计算和驱动程序的详细信息。

本应用笔记仅演示了位拆裂EUART的基本特性。有关针对基于LIN的应用专门设计的增强型功能，请参见AN2059《LIN基础知识和8位PIC®单片机上实现的MCC LIN协议栈库》（DS00002059B_CN）。

附录A： 波特率与实际速率

例A-1： 使用PIC16F1718的内部振荡器（INTOSC）时的波特率比较

Fosc = 32 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	-	-	-
300	299.45	128	0.18%
600	598.67	64	0.22%
1.200	1195.81	32	0.35%
2.400	2398.08	16	0.08%
4.800	4793.29	8	0.14%
9.600	9638.55	4	0.40%
14.400	14466.55	4	0.46%
19.200	19370.46	2	0.89%
38.400	39215.69	-	2.12%

Fosc = 16 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	99.81	256	0.19%
300	299.13	64	0.29%
600	597.64	32	0.39%
1.200	1198.32	16	0.14%
2.400	2397.36	8	0.11%
4.800	4813.48	4	0.28%
9.600	9673.52	2	0.77%
14.400	14545.45	2	1.01%
19.200	19559.90	-	1.87%
38.400	-	-	-

Fosc = 8 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	99.80	128	0.20%
300	298.86	32	0.38%
600	599.21	16	0.13%
1.200	1199.04	8	0.08%
2.400	2408.19	4	0.34%
4.800	4839.69	2	0.83%
9.600	9779.95	-	1.87%
14.400	-	-	-
19.200	-	-	-
38.400	-	-	-

Fosc = 4 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	99.74	64	0.26%
300	299.63	16	0.12%
600	599.30	8	0.12%
1.200	1204.28	4	0.36%
2.400	2420.57	2	0.86%
4.800	4889.98	-	1.87%
9.600	-	-	-
14.400	-	-	-
19.200	-	-	-
38.400	-	-	-

Fosc = 2 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	99.59	32	0.41%
300	299.58	8	0.14%
600	602.14	4	0.36%
1.200	1209.92	2	0.83%
2.400	2443.49	-	1.81%
4.800	-	-	-
9.600	-	-	-
14.400	-	-	-
19.200	-	-	-
38.400	-	-	-

Fosc = 1 MHz			
波特率	实际速率	**Time0** 预分频值	误差 百分比
100	99.93	16	0.07%
300	301.15	4	0.38%
600	605.28	2	0.88%
1.200	1222.68	-	1.89%
2.400	-	-	-
4.800	-	-	-
9.600	-	-	-
14.400	-	-	-
19.200	-	-	-
38.400	-	-	-

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。

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、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严格遵守公司的质量体系流程。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。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

ISBN：978-1-5224-1948-8

11/07/16

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://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

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
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453
Tel: 1-317-536-2380

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多 Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

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

中国 - 上海
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

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

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

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

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

亚太地区

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

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

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

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

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

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 - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

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

芬兰 Finland - Espoo
Tel: 358-9-4520-820

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

法国 France - Saint Cloud
Tel: 33-1-30-60-70-00

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-67-3636

德国 Germany - Karlsruhe
Tel: 49-721-625370

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

德国 Germany - Rosenheim
Tel: 49-8031-354-560

以色列 Israel - Ra'anana
Tel: 972-9-744-7705

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

意大利 Italy - Padova
Tel: 39-049-7625286

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

挪威 Norway - Trondheim
Tel: 47-7289-7561

波兰 Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚 Romania - Bucharest
Tel: 40-21-407-87-50

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Gothenberg
Tel: 46-31-704-60-40

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820