AN237

采用8位PIC®单片机的EUSART模块实现LIN从节点

作者： *Dinesh Baria*
Microchip Technology Inc.

简介

本应用笔记介绍了适用于带有片上增强型通用同步/异步收发器（Enhanced Universal Synchronous Asynchronous Receiver Transmitter，EUSART）外设的8位PIC®单片机的LIN 2.0从节点驱动程序。EUSART在USART的基础上新增了波特率自动检测和校准、接收到间隔字符时唤醒和13位间隔字符发送等功能，从而为LIN应用提供支持。本文档还给出了如何使用MPLAB®代码配置器（MPLAB® Code Configurator，MCC）开发LIN驱动程序的示例。本应用笔记主要介绍如何设置MCC和使用MCC生成的驱动程序，以便LIN系统设计人员能够快速设计LIN应用。在本应用笔记中，使用的单片机为PIC16F1783，但用户也可以选用任意带有EUSART模块的8位PIC单片机。

本应用笔记主要针对那些对LIN规范v2.0已经很熟悉的读者。因此，并不是所有关于LIN的详细信息都会讨论到。更多信息，请参见本文档的参考资料部分。

LIN协议

LIN（局域互联网络）为汽车和电器等多类网络提供了一套低成本的总线通信方案。LIN协议针对数据通信提供了系统开发准则。LIN的主要特性包括：

- 低成本单线实现
- 单主节点、多从节点（多达16个）
- 从节点能够同步至2K至20K范围内的任一波特率
- 从节点可自动同步至主节点速率
- 数据格式与通用串行UART格式相似
- 支持数据校验和与位错误检测，确保可靠运行

LIN 2.0与LIN 1.x的兼容性

LIN 2.0规范是LIN 1.x的超集。基于LIN 2.0规范设计的节点和基于LIN 1.x规范设计的节点彼此可以通信，但也存在一些例外情况。LIN 2.0主节点无法向LIN 1.x节点请求以下新的LIN 2.0功能：

- 增强型校验和
- 重新配置和诊断
- 自动波特率检测
- 响应错误状态监视

除非重新配置LIN 2.0从节点，否则其无法与LIN 1.x主节点配合工作。

应用

用户会问的第一个问题是：“这个驱动程序能在我的应用中正常工作吗？”本应用笔记的后续章节会给出这个问题的答案并帮助那些想知道答案的人快速地确定该驱动程序实现或器件是否适用于他们的应用。而判断是否适用的重要因素包括可用处理时间、资源使用和比特率性能。

处理时间

处理时间主要取决于比特率、时钟频率和代码执行速度。这就是推荐使用EUSART模块来实现LIN通信的原因。该硬件资源主要依靠硬件来进行处理，而对固件要求较低。因此，平均固件处理时间相对较短。表1给出了FOSC等于16 MHz时的平均固件处理时间近似值。

表1： 固件处理时间

比特率（bps）	固件处理时间（%）
2000	2.7
2400	3.2
9600	13
10417	14
20000	27

© 2008-2015 Microchip Technology Inc.

当LIN总线空闲（IDLE）时，驱动程序所用的处理时间甚至更少，在16 MHz时约为1%。

资源使用

需要使用PIC16F1783（或任何具有EUSART模块的其他器件）上的下列资源：

- EUSART——用于通信
- Timer1——用于总线和帧时序
- 程序存储器——885字，用于固件LIN驱动程序
- 数据存储器——22字节

比特率

驱动程序旨在实现LIN规范定义的最大比特率：20000 bps。为了确保正常检测间隔和同步字段，需使用同步的精确内部时钟。时钟频率由主节点定义。所有从节点均在帧头内的同步字段期间同步其内部时钟。根据LIN规范，对于完整的LIN帧（帧头加响应），允许的时钟偏差 < ±2%。

总结

LIN从节点驱动程序利用EUSART模块来处理大部分其他对处理要求比较高的事件，因此处理时间方面的问题不大。Timer1是所需的惟一其他资源，其中断频率与比特率相同。因此，驱动程序实际上是在后台透明地运行，不会对应用产生较大的干扰。

由于包括处理时间在内的大部分资源都是可用的，因此该驱动程序非常适合高需求、高处理时间的应用。

设置驱动程序

要设置固件并开始编译应用程序，用户可使用MPLAB代码配置器为EUSART和Timer1配置并生成模块驱动程序。用户需要将LIN驱动程序文件（附录中提供）添加到其项目中。

设置驱动程序之前，必须安装MPLAB X IDE、XC8 C编译器和MPLAB代码配置器（MCC）插件。这里给出建立项目所需的基本步骤：

1. 新建一个MPLAB X IDE项目或者打开一个现有项目。
2. 打开MCC插件工具。为此，需在MPLAB X IDE的菜单栏中，转到**Tools -> Embedded**（工具 -> 已安装工具），然后单击**MPLAB Code Configurator**（MPLAB代码配置器）。
3. 单击Project Resources（项目资源）区的**System**（系统）模块以调用Composer Area（设计区）的配置位设置GUI，可在其中设置配置位。如果使用EC/HS/XT振荡器，还需输入系统频率。请参见图1，注意黑色矩形框中的内容。
4. 从Device Resources（器件资源）区，选择Timer1和EUSART Asynchronous (NRZ)（EUSART异步（NRZ））模块。双击这两个模块后，它们会出现在Project Resources区。应用中使用的任何其他模块也都可以移至Project Resources区。
5. 在Project Resources区中，选择EUSART Asynchronous (NRZ)模块，然后按照图2所示内容对其进行配置。某些单片机（例如，PIC16F1708）具有外设引脚选择（Peripheral Pin Select，PPS）功能，允许编程人员将单片机所提供的许多外设分配至任意引脚。编程人员必须在Pin Manager（引脚管理器）窗口中锁定GPIO引脚，以将GPIO引脚配置为EUSART TX和RX引脚。
6. 在Project Resources区中，选择Timer1模块，然后按照图3所示内容对其进行配置。定时器周期值等于LIN通信所需的比特率值。如下公式可基于EUSART波特率和Timer1预分频值得出比特时间。

   T = 0xFFFF – (BRG+1) /PR

   值T表示TMR1寄存器中装载的16位值。值BRG表示波特率定时器（SPBRGH:SPBRGL）的16位值。值PR表示Timer1的预分频值。

© 2008-2015 Microchip Technology Inc.

图1： System模块GUI

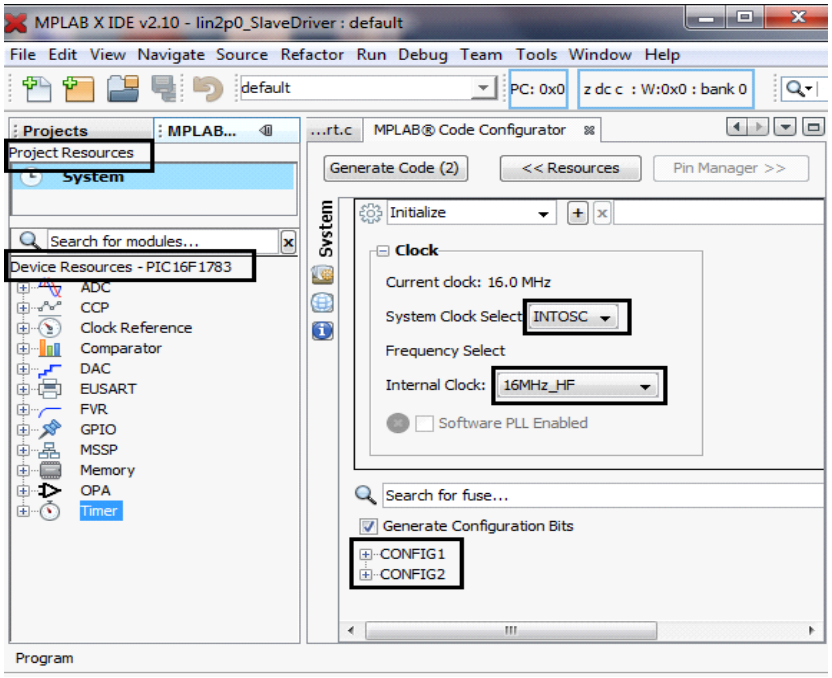


图2： EUSART Asynchronous (NRZ)配置GUI

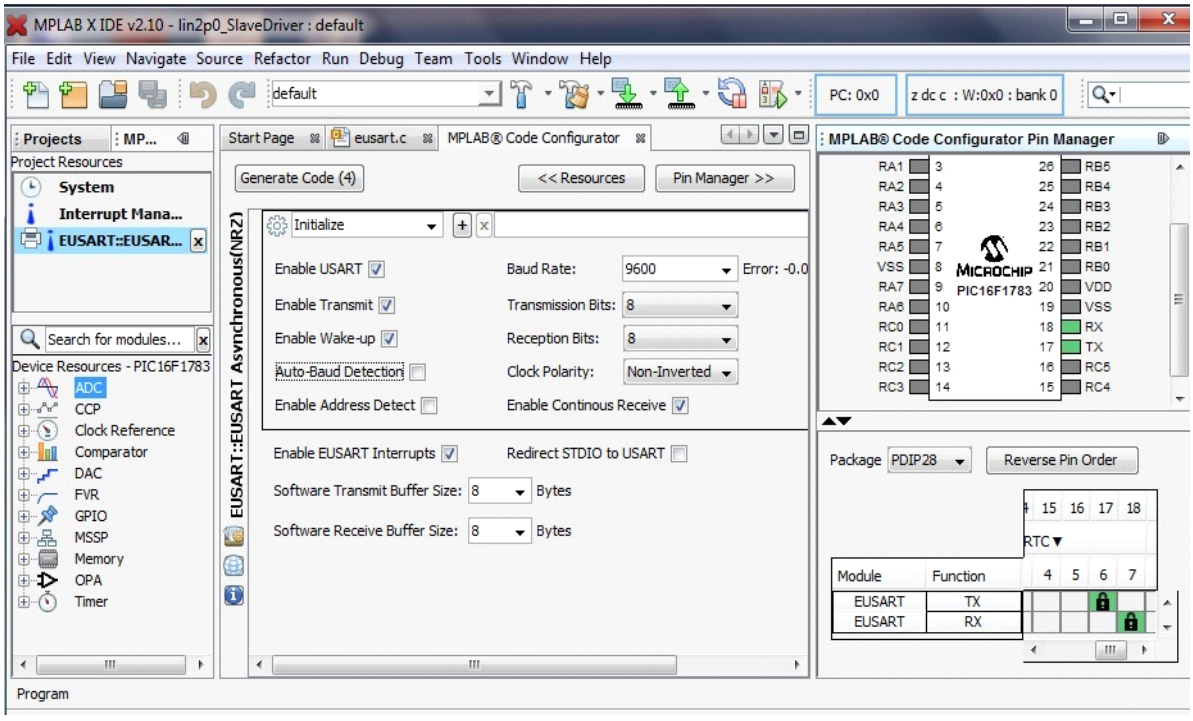
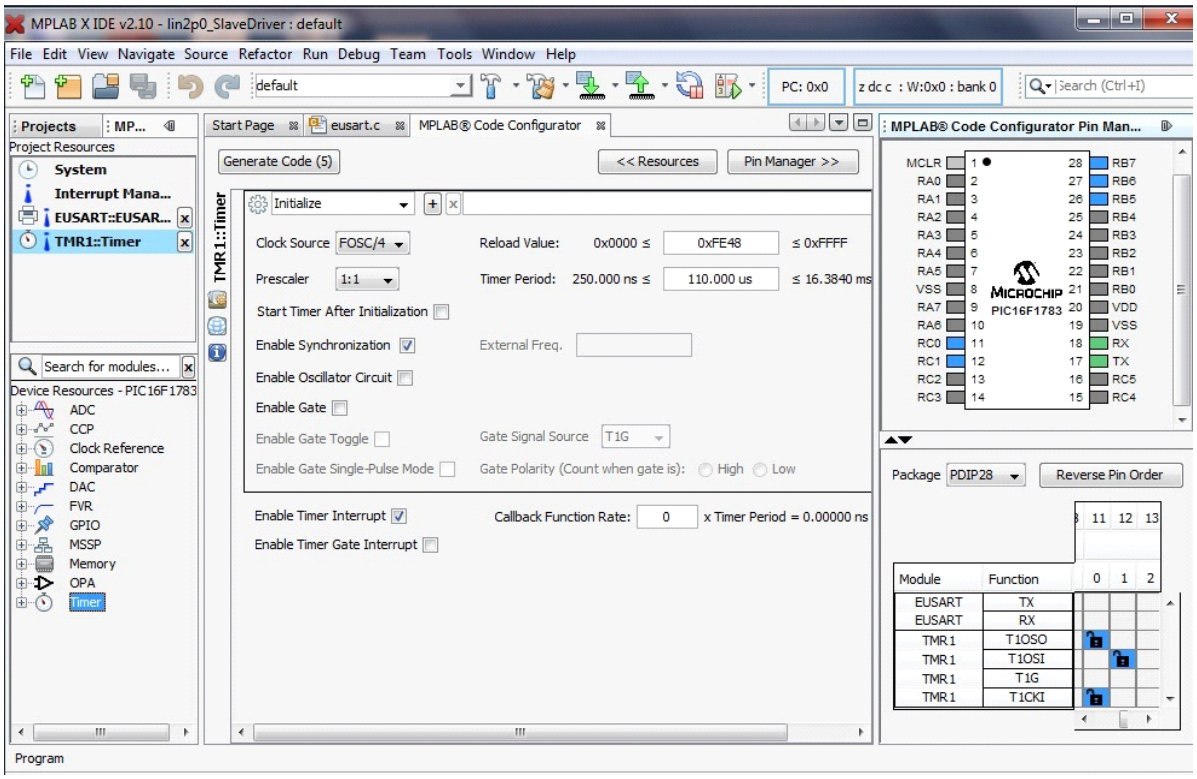
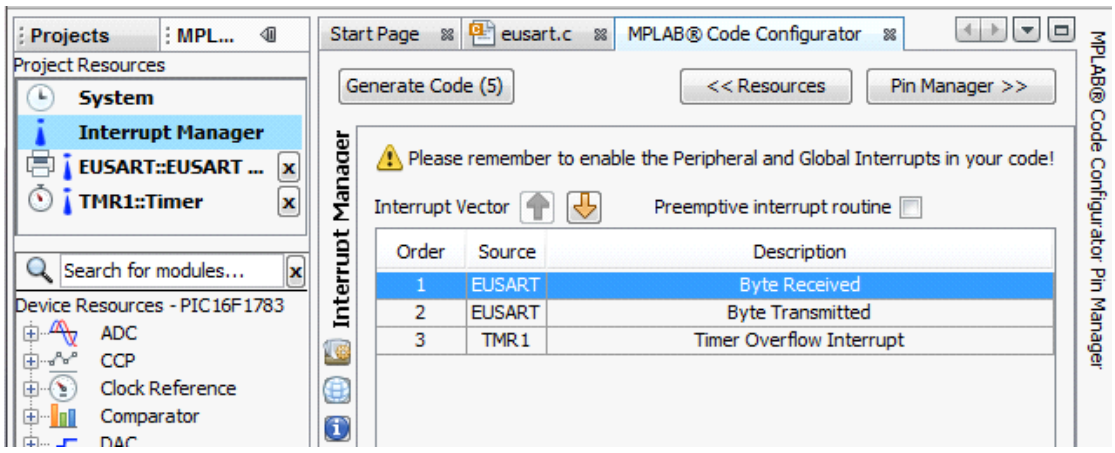


图3： Timer1配置GUI



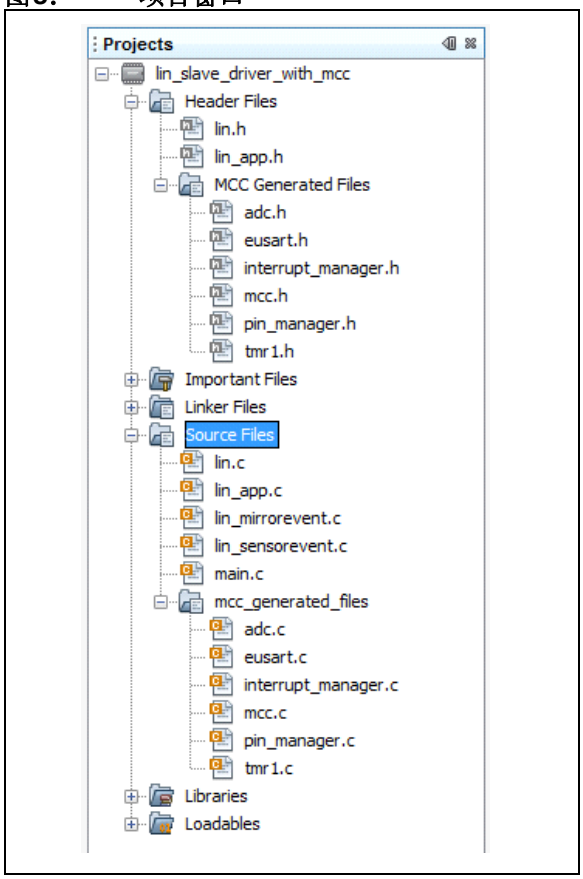
7. 然后配置移至Project Resources区的其他模块。
8. 在Project Resources区中，选择Interrupt Manager（中断管理器）模块，然后按照应用要求更改中断优先级，如图4所示。
9. 完成所有模块的配置后，单击设计区顶部的**Generate Code**（生成代码）按钮来生成驱动程序和初始化程序。生成的驱动程序将包含在活动的MPLAB X IDE项目中（见图5）。
10. 将以下LIN驱动程序文件添加到您的项目中：`lin.c`、`lin.h`、`lin_app.c`和`lin_app.h`。示例项目中提供这些文件。

图4： Interrupt Manager GUI



© 2008-2015 Microchip Technology Inc.

图5：　　项目窗口



11. `mcc.h` 和 `mcc.c` 文件包含配置位的定义和 `oscillator_initializer`。另外还包含 `system_initializer` 函数，应用程序需要调用该函数才能从内部调用其他所需的初始化程序（例如，EUSART 初始化程序和 Timer1 初始化程序）。
12. `pin_manager.h` 和 `pin_manager.c` 文件包含基于在 Pin Manager GUI 中所进行配置的引脚管理器初始化函数。
13. `interrupt_manager.h` 和 `interrupt_manager.c` 文件包含中断初始化程序和中断服务程序。
14. `tmr1.h` 和 `tmr1.c` 文件包含 Timer1 配置函数。
15. `eusart.h` 和 `eusart.c` 文件包含 EUSART 配置函数。
16. `lin.h` 和 `lin.c` 文件包含 LIN 配置函数。`lin.h` 包含新类型的定义以及 `lin.c` 中所定义的函数的原型。
17. `lin_app.h` 和 `lin_app.c` 文件是应用接口，应由用户填写。编辑这两个文件以响应相应的 ID。可以是一个表或一些简单的比较逻辑。通过这种方式，用户可定义其应用的 LIN 通信。

© 2008-2015 Microchip Technology Inc.

18. 添加其他所有与应用相关的文件。本示例中使用的lin_xxxevent.c包含与特定ID相关的应用相关功能。
19. main.c文件包含程序的入口点。根据应用需要编辑该文件。确保正确地设置中断并对驱动程序进行初始化。同时，确保包含所有外部符号。附录里提供了一个将与LIN驱动程序编译在一起的完整应用程序，以供参考。

项目

第一步是在MPLAB X IDE中建立项目。图5给出了建立项目时的项目窗口示例。LIN驱动程序运行时所需的文件如下：

- lin.c——驱动程序文件
- lin.h——用于驱动程序配置的头文件
- lin_app.h——节点配置文件
- lin_app.c——LIN事件处理文件

任何其他文件都是由系统设计人员针对特定应用定义的。例如，图5列出了这些项目文件，即lin_xxxevent.c，其中xxx表示LIN的ID事件名称。这仅是一种将ID处理按模块分配到各个单独文件的程序设计方式，从而使得项目格式更易于理解。

主文件

main.c模块中包含程序的入口点，驱动程序、硬件和变量在这里进行初始化。为了初始化LIN驱动程序，需调用LIN_Initializer函数。

在主函数中，应调用驱动程序函数LIN_Driver，如例1所示。在该函数中，将读取接收缓冲区并处理数据。

例1：　主入口点

```
void main (void)
{
    //系统初始化

    //LIN驱动程序初始化
      LIN_Initializer( );
     while(1)
     {
            //检查接收的数据
       if(EUSART_DataReady)
       {
           LIN_Driver( );
       }
       //在此添加应用程序代码
      }
}
```

修改eusart.c中的EUSART_Receive_ISR ()函数以使能自动波特率功能，如例2所示。

例2：　EUSART RX ISR

```
void EUSART_Receive_ISR(void)
{
    //溢出检查代码
    //LIN实现

    if(!EUSART_HasBreakDetected())
    {
        while(!EUSART_HasBreakDetected());
        BAUD1CONbits.ABDEN = 1;
    }
   //读取缓冲区代码
}
```

在eusart.h中，添加以下宏来使能自动唤醒功能并读取STATUS寄存器。

```
#define EUSART_Status( )        (RC1STA)
#define EUSART_EnableAutoWakeUp( )
                        (BAUD1CONbits.WUE = 1)
#define EUSART_HasBreakDetected( )
                        (! (BAUD1CON & 0x02))
```

计时函数也放在tmr1.c文件的“TMR1_ISR ()”函数中。示例固件使用Timer1来检测总线超时，但LIN设计人员可以选择任意定时器。例3给出了计时函数在中断函数内的位置。

例3：　TMR1 ISR

```
Void TMR1_ISR(void)
{
     //清零TMR1中断标志
     PIR1bits.TMR1IF = 0;

     TMR1 += timer1ReloadVal;

     //添加您的TMR1中断定制代码
     LIN_UpdateTimer( );
}
```


© 2008-2015 Microchip Technology Inc.

定义

有一些用来对系统进行设置的编译时定义，它们都位于lin.h中。表2列出并描述了这些定义。

表2： 编译时定义

定义名称	值	说明
MAX_IDLE_TIME	25000	该值是总线的最大空闲时间。LIN规范将其定义为25000。
MAX_HEADER_TIME	39	该值是最大允许帧头时间。规范将其定义为49；但时序是从第一个字节（间隔）之后开始的，因此实际上是39（比定义小10）。
MAX_TIME_OUT	128	这是发出唤醒请求之后，允许等待的最大时间。

LIN事件

LIN事件函数对ID进行解码以决定下一步操作：发送的内容、目标接收器件以及要发送/接收的字节数。设计人员应编辑或修改事件函数以处理特定的LIN ID。一种方法是建立ID表，这对需要响应多个ID的应用很有用。另一种方法是建立一些简单的比较逻辑。附录中提供的示例代码基于比较逻辑。

ID模块

示例固件使用单独的ID文件来分别处理各个ID及其相关函数。最重要的是要包含所有用到的外部符号。驱动程序所使用的符号包含在lin_app.h中，因此每个应用程序模块都应该包含lin_app.h。

lin_app.c中的函数用于设置驱动程序以处理ID事件。用户可添加该函数来处理其他ID事件。

驱动程序的使用

在使用LIN驱动程序所必需的文件建立项目之后，即可使用该驱动程序。这一部分将介绍如何使用驱动程序的相关信息。这些重要信息包括：

- 处理通信标志
- 处理错误标志
- 驱动程序内的状态标志
- LIN ID事件
- 总线唤醒

所提供的源代码是在应用中使用LIN驱动程序的一个简单示例。

通信标志

有两个标志用于指示何时成功完成数据的发送或接收。当数据被无错误地发送或接收之后，完成标志将置1。在处理完完成标志后，该标志必须由用户清零。同样，挂起标志用于指示正在进行发送或接收操作。在处理完挂起标志后，该标志必须由用户清零。

错误标志

当不满足期望的条件时，相应的错误标志将置1。例如，如果从节点接收的PID与计算的PID不匹配，则驱动程序中的奇偶校验错误标志将置1。在被处理和清除之前，错误被认为是灾难性的。因此，如果不清除错误，则驱动程序将忽略传入的数据。

例4所示的代码说明了如何在主程序循环中处理错误。该示例仅给出了对总线超时错误的响应。这一概念也可应用到其他类型的错误中。

例4： 错误处理

```
void main (void)
{
    //系统初始化

    //LIN驱动程序初始化
      LIN_Initializer( );
     while(1)
     {
            //检查接收的数据
       if(EUSART_DataReady)
       {
           LIN_Driver( );
       }
       //检查总线超时错误
       if(l_errflgs_b.bits.LE_TIMEOUT==1)
       {
            LIN_GotoSleep();
```

驱动程序状态

LIN驱动程序使用不同的状态来记住其在所接收字节之间的位置。在接收到一个字节后，驱动程序使用这些状态来决定下一个未执行状态，然后切换至该状态。

ID事件和函数

每一个ID都对应有一个事件函数。事件函数用来告诉驱动程序如何对ID之后的数据进行响应。例如，驱动程序需要准备接收或发送数据，以及需要接收或发送多少数据。

要使操作成功，必须初始化三个变量：

- 数据存储器指针
- 帧时间
- 数据长度

唤醒总线

LIN总线唤醒函数`LIN_SendWakeupSignal`可使应用具有唤醒总线的能力。调用该函数将广播唤醒请求字符。

实现

在控制LIN接口操作的相关示例固件中，可以找到以下四个函数：

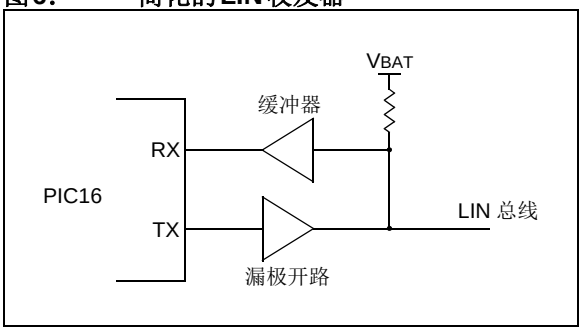
- LIN发送/接收驱动程序
- LIN计时程序
- LIN初始化
- LIN唤醒

驱动程序

增强型USART模块是用于LIN通信的关键元素。使用EUSART模块作为LIN的串行引擎具有一定的优势。其中一个显著的优势是它采用硬件进行串行控制，而不是用软件。因此，当进行数据发送或接收的同时，可以进行其他多种处理。因此，从节点LIN协议驱动程序基本上作为daemon（后台程序）在后台运行。

该驱动程序是中断驱动的，由EUSART接收中断驱动。由于LIN总线的物理反馈特性（图6），因此无论进行发送还是接收操作，都会发生EUSART接收中断。此外，错误标志用来指示发送或接收操作过程中出现的错误。

图6：　　简化的LIN收发器



© 2008-2015 Microchip Technology Inc.

状态和状态标志

LIN驱动程序使用状态标志来指示LIN收发器的当前状态。当发生中断时，驱动程序使用这些标志来决定下一个未执行的状态，然后切换至该状态。图7和图8给出了贯穿不同状态的程序流程图。表4列出并定义了这些状态。

图7：　　接收器帧头程序流程图

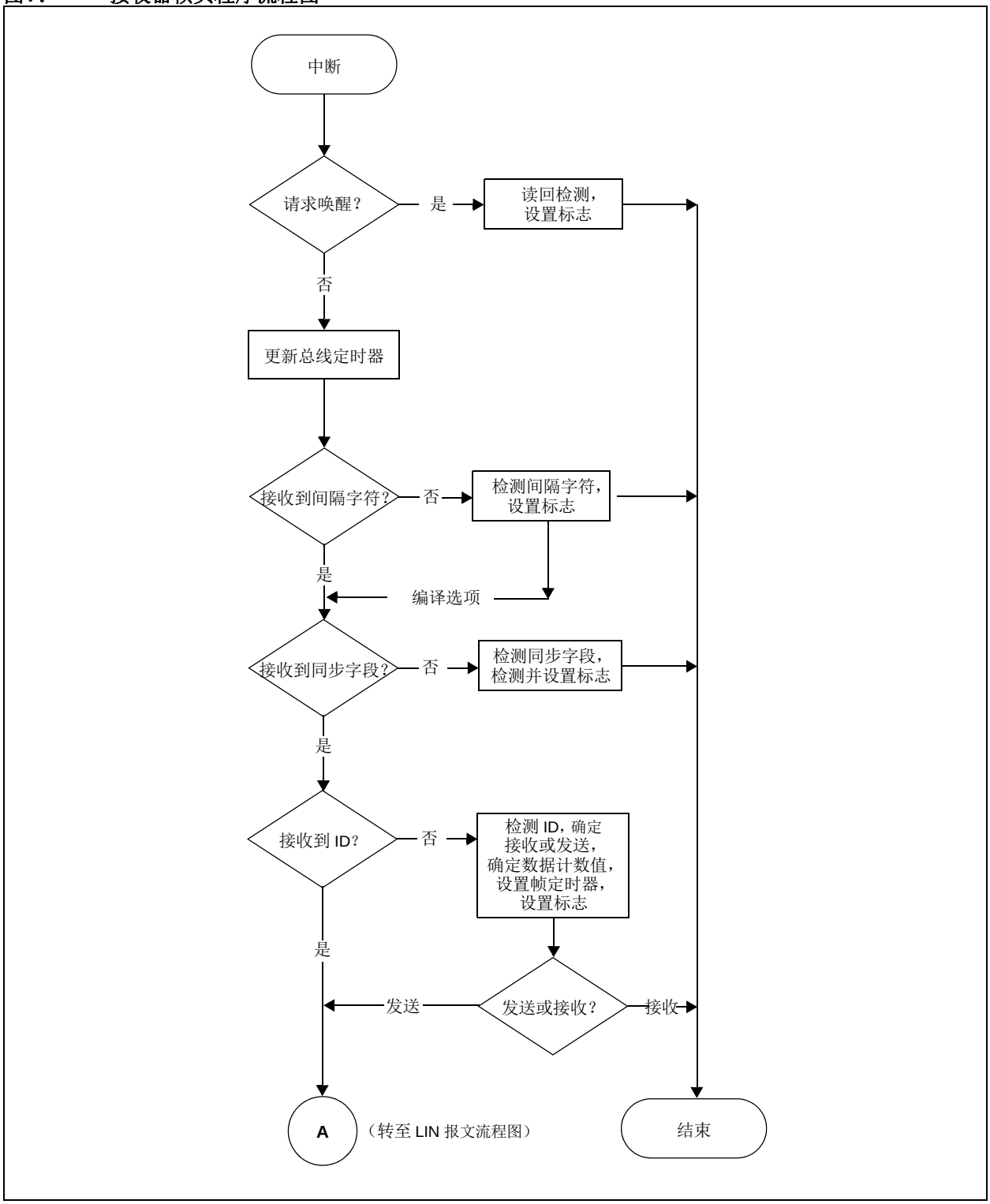
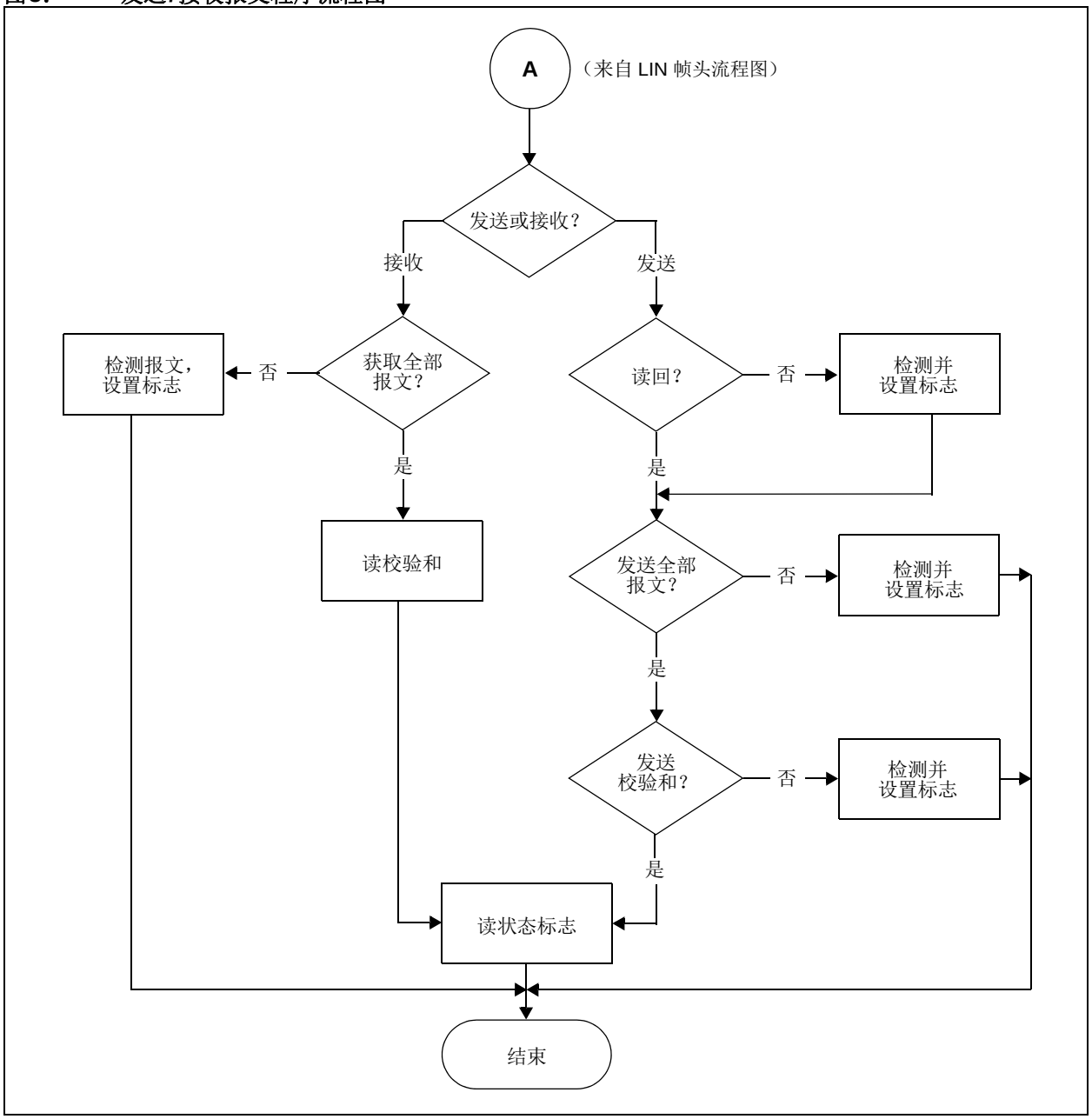


图8：　　发送/接收报文程序流程图



© 2008-2015 Microchip Technology Inc.

超时实现

超时功能为用户可选的功能，取决于用户选择的波特率。如果使能该功能，则Timer1用于跟踪允许的最大响应时间计算值。LIN规范2.0给出了帧的时隙分配要求，具体如下：

- 单位帧发送的标称值与发送的位数（不包括响应间隔、字节间隔或帧间间隔）匹配。

`THeader_Nominal` = 34 * TBit

`TResponse_Nominal` = 10*(NData+1)*TBit

`TFrame_Nominal` = `THeader_Nominal` + `TResponse_Nominal`

字节间的最大值为标称值加上40%的时间容差。

`THeader_Maximum` = 1.4 * `THeader_Nominal`

`TResponse_Maximum` = 1.4* `TResponse_Nominal`

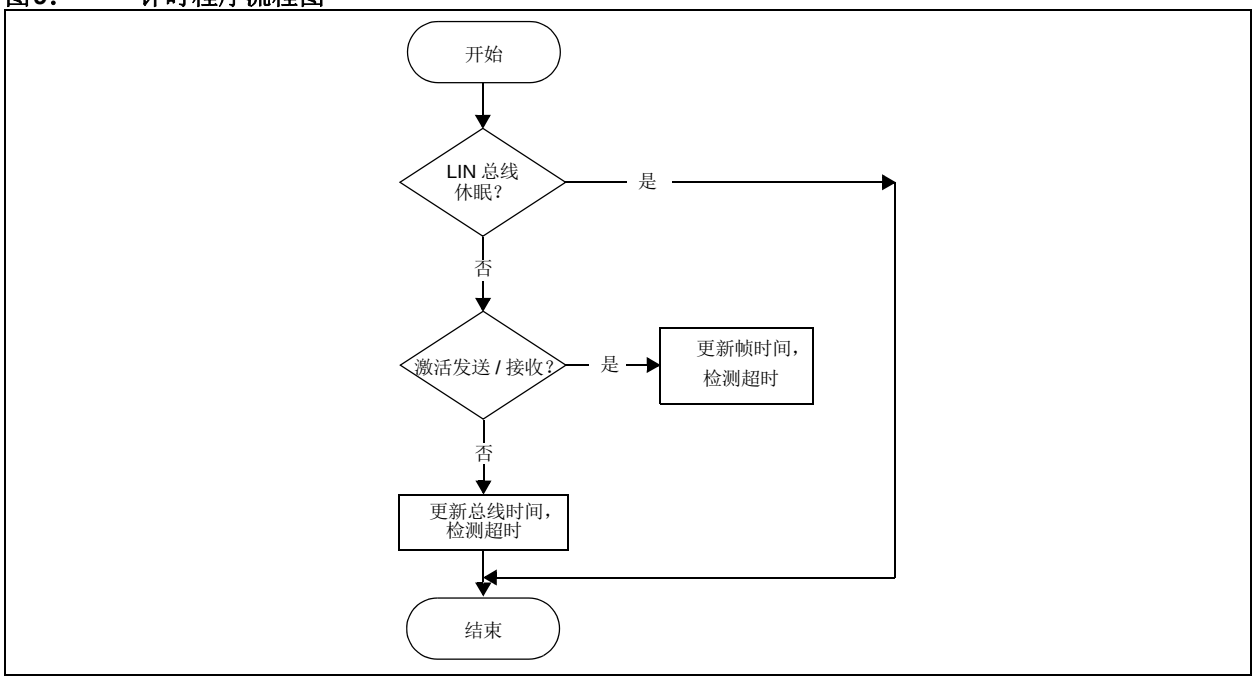
`TFrame_Maximum` = `THeader_Maximum` + `TResponse_Maximum`

- TBit是发送一位所需的时间

- NData是帧中的数据字节数

使用超时功能时，时序将从接收到前10个TBit的同步间隔后开始，使用最大帧头时间进行计算。检测到接收器缓冲区中收到0x00以及帧错误标志位验证收到同步间隔后，Timer1将使能并写入值（`THeader_Maximum` - 10*TBit）。帧头超时计数在接收到标识符或者发生超时事件后结束。标识符提供有关响应时长的信息。因此，可通过以下方式来计算响应超时值：首先用总字节（数据加一字节校验和）数乘以TBit，然后加上帧头超时未使用的剩余时间。图9给出了计时函数的流程图。

图9：　　计时程序流程图



LIN从节点驱动程序文件说明

lin.h

lin.h包含新类型的定义以及lin.c中所定义的函数的原型。

表3： 变量

变量名称	字节	说明
readback_b	1	保存将与接收数据进行比较的发送数据，用于位错误检测
chksum_w	2	驱动程序使用该变量来计算发送和接收的校验和
datalength_b	1	驱动程序使用该变量来保存报文数据的计数值
handle_b	1	用于存放接收到的标识符字节的保持寄存器
bufferptr_b	1	指向驱动程序所用存储区域的指针。将数据装入存储器还是从存储器读取数据取决于标识符
l_state_b	1	指示LIN总线所处的状态
l_errflgs_b	1	包含与LIN总线相关的错误信息
l_comflgs_b	1	指示当前的总线活动

表4： LIN状态

枚举值	状态名称	说明
0	LIN_ST_SYNCHBREAK	在该状态下，从节点准备接收间隔字段
1	LIN_ST_SYNCHBYTE	在该状态下，从节点接收同步字节
2	LIN_ST_IDENTIFIER	在该状态下，从节点接收受保护的ID
3	LIN_ST_XMIT	在该状态下，从节点响应主节点的请求
4	LIN_ST_RECEIVE	在该状态下，从节点接收来自主节点的数据
5	LIN_ST_SLEEP	在该状态下，从节点准备接收唤醒信号

© 2008-2015 Microchip Technology Inc.

寄存器1：　　错误标志

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LE_TIMEOUT	LE_NORESPONSE	LE_INCOMPLETE	LE_BITERR	LE_CHKSUMERR	LE_PARITYERR	LE_SYNCERR	LE_BRKERR
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 7　**LE_TIMEOUT：**超时错误位
1 = 发生了总线活动超时错误
0 = 未发生总线活动超时错误

bit 6　**LE_NORESPONSE：**无响应错误位
1 = 接收模式期间未收到来自主器件的响应
0 = 接收模式期间收到了来自主器件的响应

bit 5　**LE_INCOMPLETE：**未完成错误位
1 = 未完成帧的发送或接收
0 = 已完成帧的发送或接收

bit 4　**LE_BITTERR：**位错误位
1 = 通信期间出现位错误
0 = 通信期间未出现位错误

bit 3　**LE_CHKSUMERR：**校验和错误位
1 = 接收模式期间出现校验和错误
0 = 接收模式期间未出现校验和错误

bit 2　**LE_PARITYERR：**奇偶校验错误位
1 = 接收的PID中出现奇偶校验错误
0 = 接收的PID中未出现奇偶校验错误

bit 1　**LE_SYNCERR：**同步错误位
1 = 未检测到同步字段
0 = 已检测到同步字段

bit 0　**LE_BRKERR：**间隔错误位
1 = 未检测到间隔字段
0 = 已检测到间隔字段

寄存器2：　　状态标志

R-W/0-0								
LastFrame_PID<7:0>								
bit 15								bit 8

U-0	U-0	U-0	U-0	U-0	R-W/0-0	R-W/0-0	R-W/0-0	R-W/0-0
—	—	—	—	—	Goto_sleep	Overrun	Successful_transfer	Error_In_Response
bit 7								bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-8 **LastFrame_PID<7:0>：**最后一个帧的受保护标识符
最后一个帧的受保护标识符是最后一个在总线上被检测到并在节点中进行处理的受保护标识符。如果溢出标志位置1，则最后一个帧的受保护标识符将丢失一个或多个值；只会保存最新的值。

bit 7-4 **保留**

bit 3 **Goto_Sleep：**进入休眠模式标志位
1 = 自上次调用读状态函数后接收到了GOTO Sleep命令
0 = 未接收到GOTO Sleep命令

bit 2 **Overrun：**溢出标志位
1 = 自上次调用读状态函数后处理了两个或两个以上的帧
0 = 未发生溢出

bit 1 **Successful_transfer：**成功传送标志位
1 = 自上次调用读状态函数后无错误地处理了一个（或多个）帧响应
0 = 未完成帧传送

bit 0 **Err_In_Response：**响应错误标志位
1 = 自上次调用读状态函数后，由节点处理的一个（或多个）帧出现了帧响应错误（例如，校验和错误以及帧错误等）
0 = 无帧响应错误

© 2008-2015 Microchip Technology Inc.

寄存器3：　　通信标志

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	LF_COMPLETE	LF_PENDING
bit 7							bit 0

图注：		
R = 可读位	W = 可写位	U = 未实现位，读为0
-n = POR时的值	1 = 置1	0 = 清零

bit 7-2　**保留**

bit 1　**LF_COMPLETE：**通信完成标志位
1 = 已完成发送或接收
0 = 尚未完成发送或接收

bit 0　**LF_PENDING：**通信挂起标志位
1 = 正在进行发送或接收
0 = 尚未开始发送或接收

lin.c

lin.c文件包含LIN从节点操作所需的所有函数。本部分将介绍每个API函数及其相关的使用方法注释。

- *LIN_Initializer*
  - **C原型：**`void LIN_Initializer(void);`
  - **说明：**初始化LIN从节点所需的所有参数。系统初始化后，在主函数中调用该函数以启动LIN操作。
  - **参数：**无
  - **返回值：**无
- *LIN_Driver*
  - **C原型：**`void LIN_Driver (void);`
  - **说明：**该函数接收LIN信号并根据当前的LIN信号更改状态。
  - **参数：**无
  - **返回值：**无
- *LIN_ReadStatus*
  - **C原型：**`lin_status LIN_ReadStatus (void);`
  - **说明：**调用该函数将返回16位状态字。
  - **参数：**无。
  - **返回值：**16位状态字
- *LIN_SendWakeupSignal*
  - **C原型：**`void LIN_SendWakeupSignal (void);`
  - **说明：**在总线上生成唤醒命令。该函数在LIN总线上发送唤醒字符以模拟唤醒调用。
  - **参数：**无
  - **返回值：**无
- *LIN_GotoSleep*
  - **C原型：**`void LIN_GotoSleep (void);`
  - **说明：**将Goto_sleep标志位置1、禁止定时器并进入休眠模式。当主节点发送GOTO SLEEP命令时，使用该函数。
  - **参数：**无
  - **返回值：**无

- *LIN_AssignFrameID*
  - **C原型**：uint8_t LIN_AssignFrameID (uint8_t *l_NAD)
  - **说明**：该函数在具有地址NAD和指定供应商ID的从节点中分配帧的受保护标识符。更改的帧应指定报文ID并在调用该函数后将PID作为受保护的标识符。
  - **参数**：无
  - **返回值**：8位值
- *LIN_AssignNAD*
  - **C原型**：uint8_t LIN_AssignNAD (uint8_t *l_NAD)
  - **说明**：该函数分配所有从节点与 NAD、供应商识别代码和函数识别代码匹配的NAD（节点诊断地址）。这些节点的新NAD将成为新的NAD。
  - **参数**：无
  - **返回值**：8位值
- *LIN_ReadByID*
  - **C原型**：uint8_t LIN_ReadByID (uint8_t *l_NAD)
  - **说明**：该函数请求通过NAD选择的节点返回与ID参数相关的属性。请参见表5。

表5： ID的可能值

ID	说明
0	LIN产品标识
1	序列号
2 – 15	保留
16 – 31	报文ID 1...16
32 – 63	用户定义
64 – 255	保留

  - **参数**：无
  - **返回值**：8位值

lin_app.h

该文件提取自LIN描述文件（LIN Description File，LDF），用于说明节点配置。

本文件的结构如下：

- 帧定义
- 帧联合定义
- 用于数据交换的保留缓冲区大小

帧定义

- #define <Frame_Name>_Index value

定义索引值。该值是用户定义的递增整数。

- #define <Frame_Name>_ID value

定义帧标识符，取值范围为0至63（0x3F）

- #define <Frame_Name>_PID value

定义受保护的帧标识符。请参见“*LIN Specification Package Revision 2.0*”[1]中的附录7.2。

- #define <Frame_Name>_Len value

定义响应帧长度（不包括校验和）。

- #define <Frame_Name>_MsgID value

为从节点的每个事件定义报文标识符。

lin_app.h

该文件是应用接口，应由用户填写。通过这种方式，用户可定义其应用的LIN通信。

lin_app.c文件包含：

ID_Table变量

可通过填写该变量来定义应用必须处理的LIN帧的标识符。

© 2008-2015 Microchip Technology Inc.

该列表中的每个成员都对应一个LIN帧及其相应的标识符。每个成员的类型均为t_id_list，必须按以下方式定义：

例5：　　ID列表

```
typedef struct
{
    uint8_t *bufferptr;
    uint8_t len;
    uint8_t *idptr;
    t_msg_direction direction;
    uint16_t MsgID;
}t_id_list;
```

- **bufferptr**表示分配给帧的存储器地址。
- **len**表示相应帧的数据字节数。可以设置为0到8之间的值。
- **idptr**表示用于存储帧标识符的变量的地址。
- **direction**表示数据流的方向，其类型为t_message_direction，因此应设置为ID_REQUEST（对于主节点请求的数据）和ID_RESPONSE（对于从节点发送的数据）。
- **MsgID**表示每个帧的报文标识符。

参考资料

1. *“LIN Specification Package Revision 2.0”*，LIN联盟，http://www.lin-subbus.org
2. 《MPLAB®代码配置器用户指南》（DS40001725B_CN），http://www.microchip.com/pagehandler/en_us/devtools/code_configurator/home.html
3. 《MPLAB® XC8 C编译器用户指南》（DS50002053D_CN），Microchip Technology, Inc.

附录A：　　源代码

出于篇幅考虑，本应用笔记没有给出完整的源代码。您可以从Microchip网站下载完整版源代码以及所有需要的支持文件的Zip归档文件：www.microchip.com。

注：

© 2008-2015 Microchip Technology Inc.

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。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 dsPIC、 FlashFlex、 flexPWR、 JukeBlox、 KEELOQ、 KEELOQ 徽标、 Kleer、 LANCheck、 MediaLB、 MOST、 MOST 徽标、 MPLAB、 OptoLyzer、 PIC、 PICSTART、 PIC32 徽标、 RightTouch、 SpyNIC、 SST、 SST 徽标、 SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

The Embedded Control Solutions Company 和 mTouch 为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、BodyCom、chipKIT、chipKIT 徽标、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 ECAN、 In-Circuit Serial Programming、 ICSP、 Inter-Chip Connectivity、 KleerNet、 KleerNet 徽标、 MiWi、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 MultiTRAK、 NetDetach、 Omniscient Code Generation、 PICDEM、 PICDEM.net、 PICkit、 PICtail、 RightTouch 徽标、 REAL ICE、 SQI、 Serial Quad I/O、 Total Endurance、 TSHARC、 USBCheck、 VariSense、 ViewSpan、 WiperLock、 Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2008-2015, Microchip Technology Inc. 版权所有。

ISBN：978-1-63277-423-1

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

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。

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

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

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

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

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 1-905-673-0699
Fax: 1-905-673-6509

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

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

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: 86-21-5407-5533
Fax: 86-21-5407-5066

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

中国 - 深圳
Tel: 86-755-8864-2200
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 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-213-7828

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

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

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

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

德国 Germany - Pforzheim
Tel: 49-7231-424750

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

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

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

波兰 Poland - Warsaw
Tel: 48-22-3325737

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

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

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

01/27/15

© 2008-2015 Microchip Technology Inc.