AN729

使用 PIC® MCU 实现 LIN 协议

作者: *Dan Butler*
Thomas Schmidt
Thorsten Waclawczyk
Microchip Technology Inc.

介绍

LIN 协议由欧洲汽车制造商协会设计，是一种低成本、短距离的低速通信网络。它旨在传送开关设置的变化，并对开关变化做出响应，因此通信事件是在人类所能感知的几百毫秒时间内发生的。

本应用笔记并非要取代或者重建LIN协议规范。它只是对该总线进行了全面介绍，并从一个较高的角度来说明总线工作原理、如何基于 PIC® 器件实现从节点及其功能。可从 www.lin-subbus.com 网站获得完整的 LIN 协议规范。然而，在撰写本应用笔记时，LIN 协议规范的副本仅由 Audi AG、BMW AG、DaimlerChrysler AG、Motorola, Inc.、Volcano Communication Technologies AB、Volkswagen AG 和 Volvo Car Corporation 发布。

总线特性

LIN 协议支持在单根线上进行双向通信。它采用廉价、由 RC 振荡器驱动的单片机，这样可以省去晶振或陶瓷谐振器的成本。此协议实际上是以时间和软件上的代价换取精密硬件上成本的节约。该协议的每一条报文都包含自动波特率调节的过程。支持最高 20K 波特率的传输速率，以及低功耗休眠模式，休眠模式下总线被关闭以避免消耗电池，然而总线可由总线上的任一节点进行供电。

LIN总线融合了I^2C^{TM}和RS232的特性。与I^2C类似，LIN总线通过电阻拉为高电平，每一个节点通过一个集电极开路驱动器拉为低电平。然而，由于没有时钟线，故又像 RS232 那样，通过起始位和停止位来标识每个字节，而各个位采用异步时序进行传输。

电气连接

图 1 显示了典型的 LIN 协议配置。

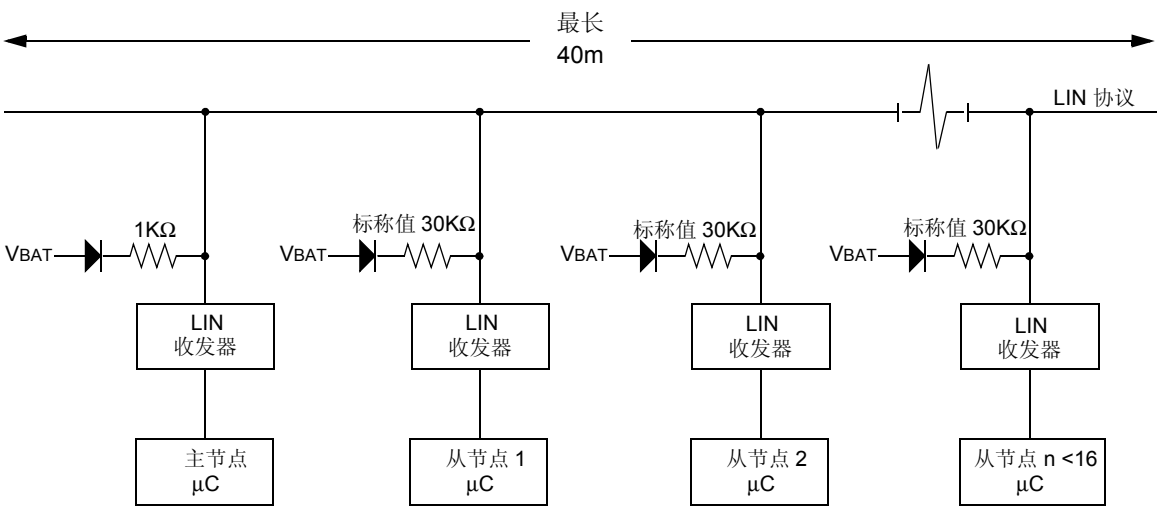
总线使用单根线进行通信，这根线通过电阻和集电极开路驱动器拉为高电平。当任何一个节点将总线拉低时，总线处于地电平，标志着总线进入显性状态。当总线电平为 V_{BAT}（9–18V）且要求所有节点都使总线处于悬空状态时，则意味着总线处于隐性状态。在空闲状态下，总线通过上拉电阻悬空为高电平。

总线工作电压为 9V 至 18V，但总线上的器件必须能够承受 40V 的电压。通常，单片机通过线路驱动器 / 接收器与总线电平隔离。这样能使单片机工作于 5V 电压条件下，而总线工作于更高的电压。

总线上的每个节点都端接到 V_{BAT}。主节点通过 1 KΩ 电阻端接，而从节点通过 20–47 KΩ 电阻端接。总线最大长度设计为 40 米。

在本应用笔记截稿（2000 年初）时，将使用 K 线路驱动器直至出现真正的 LIN 驱动器。

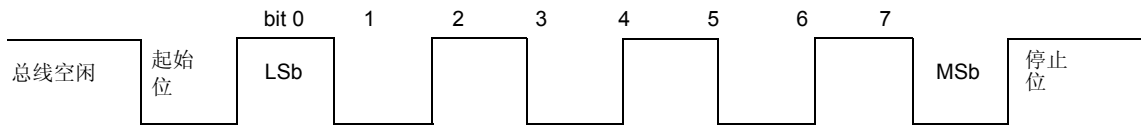
图 1：　　总线配置



字节协议

如图 2 所示，每个字节都与起始位和停止位一起组成帧。在每个字节中，数据从最低位开始发送。起始位与空闲状态相反（为 0），停止位与空闲状态相同（为 1）。

图 2：　　字节协议



每个字节都与起始位和停止位一起组成帧，数据从最低位开始发送。

报文协议

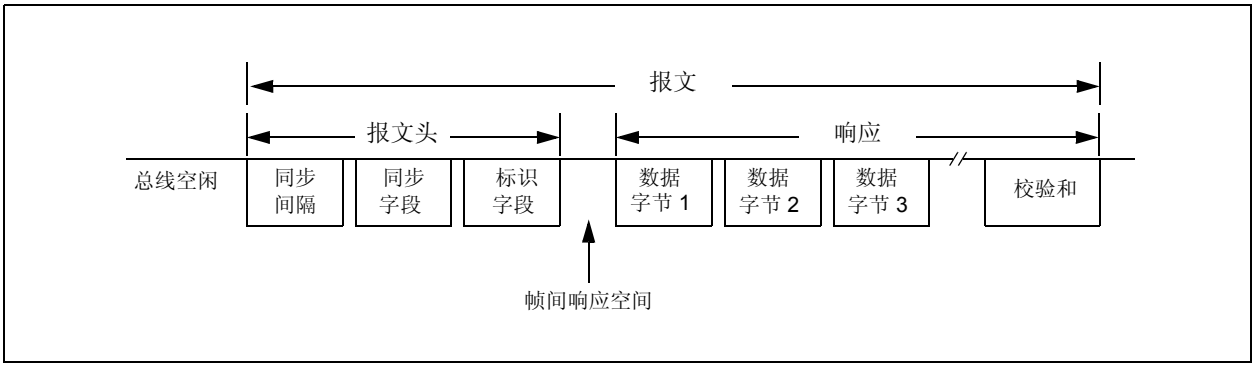
主节点通过查询从节点实现与总线其余部分的数据共享。从节点仅在接收到主节点命令后才进行数据发送，从而在无须进一步仲裁的情况下进行双向通信。报文传送以主节点发送同步间隔开始，随后发送同步字段和报文字段。主节点通过在每条报文的起始发送同步字段还为整个总线设定了时钟，以此实现时钟同步。每个从节点都必须使用这一同步字节对其波特率进行调整。

同步间隔使总线进入显性状态，该状态保持 13 个位时间，紧接着是一个停止位（隐性状态），这将告知从节点即将有报文传输。主节点与从节点的时钟漂移最大为 15%。因此，从节点接收到的同步间隔可能仅有 11 个位时间或长达 15 个位时间。

每条报文的第二个字节是标识字节，用于告诉总线随后将发送什么数据，并指明哪个节点应该应答以及应答的长度。只有一个从节点可以响应给定的命令。

从节点仅在主节点的控制下才能在总线上发送数据。一旦数据出现在总线上，任何节点都可以接收该数据。因此，从节点之间的通信不需要主节点进行控制。

图 3：　　报文协议



时钟同步

LIN协议使用控制器中的低成本RC振荡器。为了使每个节点在时钟出现漂移的情况下仍能正常工作，从节点必须在每次传输时检测主节点的波特率并据此调整自身的波特率。因此，每一个事务都由同步字段开始。同步字段是一个字节 0x55（交替的 0 和 1）。这使得每个从节点能检测 8 个位时间。通过计算这些跳变、将其除以 8 并作取整处理，使得每个从节点根据主节点调整自身的时序。

标识字段

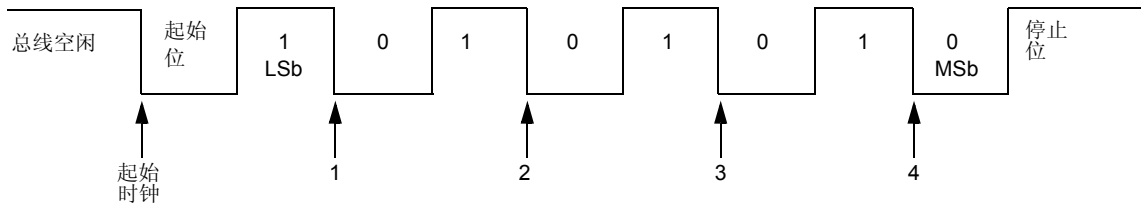
同步字段之后是标识字段，用于告知总线接下来传送的内容是什么。标识字段又被分为 3 个字段：最低 4 位（bit 0–3）用来对总线上的器件进行寻址；中间2位（bit 4–5）指示此后报文的长度；最后 2 位（bit 6–7）用于奇偶校验。

标识字段的 4 个地址位可以寻址最多 16 个从节点，每个从节点又可发送 2、4 或 8 字节响应，这样总共有 64 种不同的报文。

除了休眠命令 （详见 “低功耗休眠模式”章节）外，LIN 协议规范并没有定义每条报文的内容。而是留在具体应用程序中定义。

节点间可直接传送报文，如同在主节点控制下一样。数据无须被主节点接收后再重新发送到接收节点。相反，任何报文都可被任何节点接收并处理。

图 4： 同步字段



计数时间为 4 个连续的下降沿，然后除以 8 并取整以获得单个位时间。除操作和取整操作可通过将数据右移 3 位 （带进位）方便地实现。

图 5：　　标识字段

$\overline{P1}$	P0	ID5	ID4	ID3	ID2	ID1	ID0

P0：　　奇偶校验位ID0 ⊕ ID1 ⊕ ID2 ⊕ ID4

$\overline{P1}$：　　奇偶校验位ID1 ⊕ ID3 ⊕ ID4 ⊕ ID5

ID0 – 3：　　器件地址

ID4 – 5：　　报文长度

ID5	ID4	数据字节
0	0	2
0	1	2
1	0	4
1	1	8

错误检测

每个节点中都要对以下错误进行检测和计数：

- 位错误：正在发送的节点应将期望发送的数据与总线上实际出现的数据进行比较。控制器必须等待足够长的时间以确保总线对发送的数据作出响应，然后才可以测试该位。假定最小边沿斜率为 1V/µs 而最大总线电压为 18V，那么发送器应等待 18 µs 后才能进行测试，以确定总线上该位是否正确。
- 校验和错误：每条报文所包含的数据内容都由校验和字节保护，校验和字节是数据字节相加后取 256 模的余数再取反的结果。
- 奇偶校验错误：命令字节用 2 个奇偶校验位来保护其他 6 位。这需要重新计算和比较。

如果有错误，命令将被忽略，同时记录错误。

错误报告

不存在直接错误报告机制。然而，每个从节点都应能跟踪自己的错误。这样主节点可请求将错误状态作为正常报文协议的一部分。

CAN 总线接口

LIN 协议并不能直接与 CAN 总线兼容，然而我们希望它们能进行相互操作。 CAN 总线可用于整个汽车内的通信，而 LIN 协议仅用于汽车各个模块内部的通信，如车门内部。

为了连接这两个总线，需要采用 CAN-LIN 协议接口节点。接口节点负责从 LIN 协议节点收集信息然后将信息传送到 CAN 总线上。

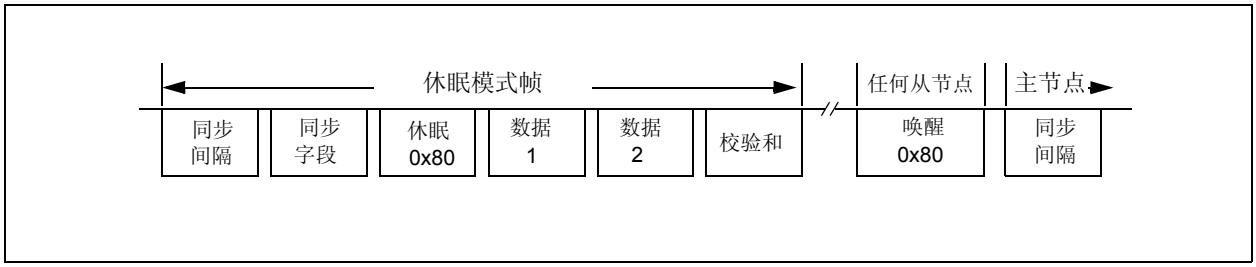
低功耗休眠模式

主节点通过发送标识码0x80使所有节点进入休眠模式。这是 LIN 协议规范中定义的唯一报文 ID。休眠命令后的数据字节内容没有定义。接收到休眠命令的从节点应当对自身进行设置，以便当总线状态发生改变时能唤醒，并关闭自身的电源以使电流消耗最低。总线悬空为高电平且不消耗电流。

任何节点都可通过发送唤醒信号来唤醒总线。该信号为字符0x80（7位低电平时间和其后的1位高电平时间）。接收到此信号后，所有节点应唤醒并等待主节点开始以标准方式查询总线。

如果主节点未能在 128 个位时间 （波特率为 20K 时为 6400 us ）后唤醒，那么试图唤醒主节点的节点应再次尝试。在等待 15000 个位时间 （波特率为 20K 时为 750 ms ）之前，总共可尝试 3 次。

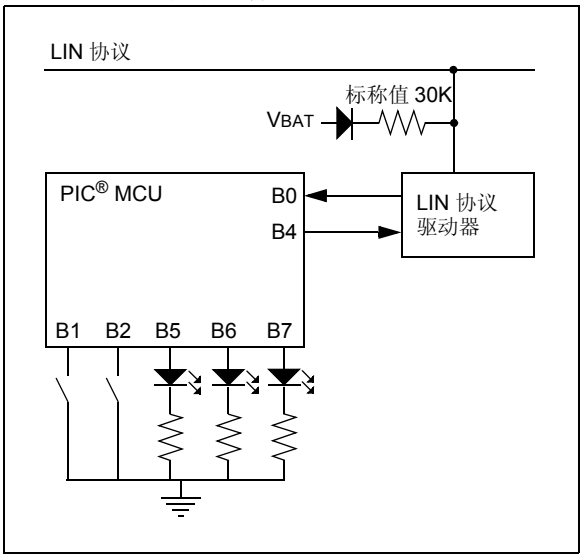
图 6： 休眠报文



演示软件

附录 A 中的代码演示了 LIN 协议通信。如图 7 所示，硬件包括 2 个按钮和 3 个 LED。每按 10 次按钮 1， LED 1 将改变一次状态。同样，每按 10 次按钮 2，LED 2 将改变一次状态。响应 ID1，按钮计数值将被发送到总线上。响应 ID4，通过总线更新按钮计数值。

图 7： 演示硬件



软件操作

LIN协议代码基于由RB0触发的中断进行工作, 以实现休眠 / 唤醒功能。一旦触发中断，将对低电平位时间的长度进行计数。然后读同步字节并确定本地位时间。最后将其与原始位时间作比较，如果原始低电平时间大于 10 个位时间，则发出同步间隔信号；若小于 10 个位时间，则发出从休眠模式唤醒的信号。

如果是从休眠模式唤醒，则代码退出并继续等待同步间隔。

如果是同步间隔，则读入命令字节、检查奇偶校验位并检查操作表以确定接下来的操作。操作表定义了总线上数据的来源及目的地。

软件功能

为了初始化 LIN 协议从节点处理程序，用户必须调用程序 `InitLinSlave`。此程序将初始化 RB0 中断引脚和 TMR0。 TMR0 用来测量位长度并生成波特率。初始化完成之后，用户可以执行自己的代码。一旦检测到 RB0 上的下降沿，代码将中断并跳转到中断服务程序执行。必须禁止除 TMR0 和 RB0 外的所有中断以精确计时同步字段。计算出波特率后，退出中断服务程序。下一次 RB0 中断到来时，LIN 协议从节点处理程序自动进入接收模式来接收标识字段或数据字节。一旦检测到标识字段或数据字节的起始位 （此时程序跳转到中断服务程序），就将接收标识字段并对其进行解码。代码执行取决于接收到的标识字段，如存储数据、点亮 LED 等等。程序执行后，用户必须将该代码包含在子程序 `DecodeIdTable` 中。

总线帧完成后，将标志 `FCOMPLETE` 置 `1`。此标志指示所有数据已正确接收并可以进行后续处理。此标志必须由用户固件清零。

注： TMR0 用来测量位时间并生成波特率。因此，应用软件不能对 TMR0 进行操作。

错误检测

从节点进程对以下错误进行检测：

- 校验和错误
- 位错误
- 停止位丢失
- 奇偶校验错误
- 超时错误

从节点检测到错误时，错误将被记录到ERRORFLAGS 寄存器中，如图 8 所示，并将忽略接收到的报文。一旦接收到有效的标识字段，这些错误标志将被清零。

图 8：　　错误标志

7	6 . . . 4	3	2	1	0
超时	未用	位错误	CRC 错误	ID 奇偶校验错误	无响应

当检测这些错误时，相应的标志将被置 1。

软件性能

LIN 协议从节点处理程序的工作速度最高达 20K 波特率。

LIN 协议从节点处理程序需要占用 420 字的程序存储空间（不包括 In Out ID 的宏所占用的程序存储空间）和 23 字节的数据存储空间。

本应用程序非常适用于工作频率为 4 MHz 的 Microchip 单片机内部的 RC 振荡器。

集成到用户代码中

用户必须对子程序 `DecodeIDTable` 进行编辑。在这个子程序中，用户将对接收到某个标识字段时所采取的操作进行定义。而且，用户也可对出现某些错误（即时序错误或其他错误）时所采取的操作进行定义。

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 PIC® 单片机产品的软件。

本软件为本公司和 / 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。

附录 A： 源代码

```
MPASM 02.30.09 中间版本 LINSLAVE.ASM   1-27-2000  9:57:59          第 1 页


LOC  目标代码          行号      源代码
  值

                     00001 ;                                软件许可协议
                     00002 ;
                     00003 ; 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
                     00004 ; 用于 Microchip 生产的 PIC® 单片机产品的软件。
                     00005 ;
                     00006 ;
                     00007 ;
                     00008 ; 本软件为本公司和 / 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
                     00009 ; 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。
                     00010 ;
                     00011 ;
                     00012 ;
                     00013 ;
                     00014 ; 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
                     00015 ; 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
                     00016 ; 的损害，本公司概不负责。
                     00017 ;
                     00018 ;
                     00019 ;
                     00020 ; ********************************************************************************
                     00021 ;       文件名：                        LINSLAVE.ASM
```

```
00022 ;                                  LIN （局域互连网）规范 1.0 版
00023 ;
00024 ; ###############################################################################
00025 ;
00026 ;                 作者:               Thorsten Waclawczyk
00027 ;                 公司:               Arizona Microchip Technology GmbH
00028 ;
00029 ;                 版本:               1.7
00030 ;                 日期:               2000 年 1 月 18 日
00031 ;                 使用的汇编器:       MPASM 2.30.07
00032 ;
00033 ;###############################################################################
00034 ;
00035 ;       头文件:
00036 ;       p16C622.inc     1.01 版
00037 ;
00038 ;###############################################################################
00039 ;
00040 ;       通过使用外部中断 INTE 和 TMR0 实现基于中断的系统，实现符合 LIN 规范 1.0 版的
00041 ;       LIN 从节点处理程序。
00042 ;
00043 ;       LIN 处理程序说明
00044 ;
00045 ;       必须在主任务开始之前通过调用子程序 “InitLinSlave” LIN 对从节点处理程序进行初
00046 ;       始化。
00047 ;
00048 ;       在初始化之后，基于中断的 LIN 处理程序等待第一个下降沿和第二个上升沿到来测量同步
00049 ;       间隔低电平时间的长度。
00050 ;       随后 LIN 处理程序等待下一个下降沿以对后续四个下降沿检测事件之间的处理器周期进行
00051 ;       计数。
00052 ;
00053 ;       将这一结果除以 8 以获得位长度。为确保此为报文头序列，将位长度乘以 10 并与初始同
00054 ;       步间隔计数值进行比较。如果同步间隔计数值更长，可知此为同步间隔。
00055 ;
00056 ;
00057 ;
00058 ;       此后，通过将位长度值保存到 timer0，将 LIN 处理程序设置为接收模式来读标识字段。
00059 ;
00060 ;       接收到该字节后，将在子程序 “CheckIdentifierByt” 中对其进行检测和解码，这样
00061 ;       LIN 处理程序能够处理输入或输出的响应帧。
00062 ;
00063 ;
00064 ;       LIN 处理程序将处理时序和错误检查，因此用户必须做的只是定义 DecodeIdTable。
00065 ;
00066 ;       这将给主机发出的 16 个可能 ID 分配一个操作。有以下三种可能的操作：
00067 ;       MacroLstMode 监听数据但不对其进行操作；
00068 ;       MacroRsMode 为接收数据提供缓冲区；
```

```
00069 ;       MacroTxMode 则为发送数据建立缓冲区。
00070 ;
00071 ;       当处理完一个完整的报文帧后，标志 “COMFLAGS,FCOMPLETE”将置 1。因此， 此后若没有错误，
00072 ;       用户便可以处理缓冲区中的数据。 处理完数据之后，用户必须清零此完成标志以表示数据已处理完
00073 ;       毕。
00074 ;
00075 ;
00076 ;       演示程序说明
00077 ;
00078 ;       从节点具有两个按钮和三个 LED。对按钮状态进行读操作并计数。为响应 ID1，将发送按钮计数值。
00079 ;
00080 ;
00081 ;       当对应于计数器 （COMPLED1 或 COMPLED2）的寄存器值与限定值 （当前该值设定为 10）匹配时，
00082 ;       LED 1 和 LED 2 将改变状态。
00083 ;
00084 ;
00085 ;       ID4 通过 LIN 总线对计数器进行更新。
00086 ;
00087 ;       当检测到一个完整的报文帧时， LED3 将翻转状态。
00088 ;
00089 ; ##############################################################################
00090 ;
00091 ;       更改内容
00092 ;
00093 ;       更改                    更改日期
00094 ;
00095 ;
00096 ; ##############################################################################
00097 ;
00098 ;       程序存储器和数据存储器使用
00099 ;               程序存储器    :       0x0420
00100 ;               数据存储器    :       0x24
00101 ;               堆栈层级      :       2
00102 ;
00103 ; ##############################################################################
00104
00105         LIST p=16C622,F=INHX8M
00106
00107 #include <p16c622.inc>
00001         LIST
00002 ; P16C622.INC 标准头文件， 1.01 版     Microchip Technology, Inc.
00165         LIST
00108
2007   3FFB   00109         __CONFIG   _CP_OFF & _WDT_OFF & _RC_OSC
00110
00111 ; #### 定义 ########################################################################
00112
```

```
                        00113 #define RSLINEPIN       0                       ; PB0 连接到接收线
                        00114 #define TXLINEPIN       4                       ; PB4 连接到发送线
                        00115
                        00116 #define LINSLAVERAM     0x20                    ; 起始地址
                        00117
                        00118 ; ---- LINSLAVE 模块使用的 RAM 地址单元  ------------------------------------
                        00119
                        00120         CBLOCK  LINSLAVERAM
00000020                00121         COPYWREG                                ; 保存 WREG 备份
00000021                00122         COPYSTATUS                              ; 保存 STATUS 备份
                        00123
00000022                00124         HICOUNT:0,TIMEOUTLO,TIMEOUTHI           ; TMR0 和超时计数器的临时高字节
                        00125                                                 ;
00000024                00126         COUNTEDGES                              ;
00000025                00127         COUNTVALUE:0,COUNTVALUELO,COUNTVALUEHI  ; 波特率的软件计数器
                        00128
00000027                00129         SYNCLENGTH:0,SYNCLENGTHLO,SYNCLENGTHHI
00000029                00130         BITREG:0,BITNBR,BITLENGTH               ; 位位置和位长度
0000002B                00131         DATABLOCKLENGTH                         ; 发送 / 接收数据块数
0000002C                00132         PREIDNUMBER                             ; 保存 id 字段 bit 5..0
0000002D                00133         IDNUMBER                                ; 保存 ID 值
0000002E                00134         ERRORFLAGS                              ; 协议通信错误
0000002F                00135         COMFLAGS                                ; LIN 总线通信位
00000030                00136         BUFFERPTR
00000031                00137         COMBUFFER
00000032                00138         DATACRC                                 ; 发送 / 接收数据块的校验和
00000033                00139         TXDATAFIELD:8                           ; 数据发送缓冲区
0000003B                00140         RSDATAFIELD:8                           ; 接收数据块缓冲区
00000043                00141         DUMMY:2
                        00142         ENDC
                        00143
000000A0                00144 MIRRCOPYWREG    EQU     (COPYWREG + 80h)        ; page1 中保留的 RAM 存储单元
                        00145
                        00146 #define LINSLAVEBLOCKLENGTH (DUMMY+1 - LINSLAVERAM)
                        00147 #define LINBLOCKEND     (DUMMY+1)
                        00148 ;*****************************************************************
                        00149 ;                                               ; 计算 LIN 从节点需要的 RAM 空间
                        00150
                        00151
                        00152 ; #### 位定义 #
                        00153 ; 定义错误标志变量
                        00154
                        00155 #define FTIMEOUT        7                       ; 无接收 / 发送
                        00156 #define FBITERROR       3                       ; 输出的位值
                        00157                                                 ; 与总线值不同
                        00158 #define FCRCERROR       2                       ; CRC 校验错误
                        00159 #define FIDPARITYERROR  1                       ; ID 的奇偶校验错误
```

```
                  00160 #define FNORESPONSE     0                       ; 总线上无响应信号
                  00161
                  00162 ; 定义通信标志变量
                  00163
                  00164 #define FSYNCHBREAK     0                       ; 检测到同步间隔
                  00165 #define FID             1                       ; 将 ID 字节作为下一个字节接收
                  00166 #define FRSDATA         2                       ; 读数据块
                  00167 #define FTXDATA         3                       ; 要发送的数据块
                  00168 #define FCOMDATA        4                       ; 正在进行通信
                  00169 #define FLISTENONLY     5                       ; ID 字节为仅用于监听的命令
                  00170 #define FCOMPLETE       6                       ; 通信结束
                  00171 #define FSLEEPMODE      7                       ; 休眠模式帧
                  00172
                  00173
                  00174 ; ###############################################################################
                  00175 ; #### 主任务声明 ########################################################
                  00176 ;
                  00177 ;       这里将对演示程序中使用的变量进行定义
                  00178 ;
                  00179
                  00180 #define LED1            5                       ; 按钮 1 按下 10 次后翻转
                  00181 #define LED2            6                       ; 按钮 2 按下 10 次后翻转
                  00182 #define LED3            7                       ; 指示完整的报文帧
                  00183
                  00184 #define BUTTON1         1                       ; 输入
                  00185 #define BUTTON2         2
                  00186
                  00187         CBLOCK  TXDATAFIELD
00000033          00188         BUTTONPRESSED1                          ; ID1 使用的寄存器
00000034          00189         BUTTONPRESSED2                          ;
                  00190         ENDC
                  00191
                  00192         CBLOCK  RSDATAFIELD
0000003B          00193         COMPLED1                                ; ID4 使用的寄存器
0000003C          00194         COMPLED2                                ;
                  00195         ENDC
                  00196
                  00197 ; 定义主任务使用的全局变量
                  00198
                  00199         CBLOCK LINBLOCKEND+1
00000045          00200         COMPERATOR                              ; 包含点亮 / 熄灭 LED 的限定值
                  00201                                                 ;
00000046          00202         B1FILTER                                ; 防抖滤波按钮 1

00000047          00203         B2FILTER                                ; 防抖滤波按钮 2
00000048          00204         DEBOUNCECOUNTER
00000049          00205         COPYPORTB
```

```
0000004A            00206         EDGEDETECT
                    00207         ENDC
                    00208
                    00209 ; ###########################################################################
                    00210 ;###########################################################################
                    00211
                    00212 ; #### 声明模式宏 ###################################################
                    00213
                    00214 MacroRsMode MACRO       StartAddrOfPtr
                    00215         bsf     COMFLAGS,FRSDATA              ; 指示处于接收模式
                    00216         movlw   StartAddrOfPtr                ; 获得存放数据所需的
                    00217                                               ; ram 存储单元
                    00218         movwf   BUFFERPTR                     ; 并初始化指针
                    00219         retlw   0
                    00220         ENDM
                    00221
                    00222 MacroTxMode MACRO       StartAddrOfPtr
                    00223         bsf     COMFLAGS,FTXDATA              ; 指示处于发送模式
                    00224         movlw   StartAddrOfPtr                ; 获得从中读取的所需
                    00225                                               ; ram 存储单元
                    00226         movwf   BUFFERPTR                     ; 并初始化指针
                    00227        retlw   0
                    00228        ENDM
                    00229
                    00230 MacroLstMode MACRO
                    00231         bsf     COMFLAGS,FLISTENONLY          ; 指示处于接收模式
                    00232                                               ; 但并不保存接收到的数据
                    00233         nop                                   ; 这对于获得正确的跳转表长度
                    00234         nop                                   ; 是必需的
                    00235         retlw   0
                    00236         ENDM
                    00237
                    00238 ; #### 复位向量 ###########################################################
                    00239
0000                00240 RESET   org     0x0000                      ; 复位向量
                    00241
0000   29A9         00242        goto    Main
                    00243
                    00244 ; #### INTvector ##########################################################
                    00245
0004                00246 INTvector       org     0x0004
0004   00A0         00247         movwf   COPYWREG                      ; 保存使用的寄存器 WREG
0005   0803         00248         movf    STATUS,W
0006   1283         00249         bcf     STATUS,RP0                    ; 切换到 bank0
0007   00A1         00250         movwf   COPYSTATUS                    ; 将 STATUS 的内容保存到 page0 中
0008                00251 TMR0int
0008   1D0B         00252         btfss   INTCON,T0IF
```

```
0009   2815           00253         goto    ExtInt                          ; 如果允许定时器中断
                      00254
000A   0AA2           00255         incf    HICOUNT,F                       ; 递增高字节，也是
000B   1D03           00256         btfss   STATUS,Z                        ; 超时计数器的低字节
000C   2810           00257        goto    TMR0Int1
                      00258
000D   0AA3           00259         incf    TIMEOUTHI,F                     ; 3000 * 256 个周期之后
000E   19A3           00260         btfsc   TIMEOUTHI,3                     ; 出现超时序列
000F   17AE           00261         bsf     ERRORFLAGS,FTIMEOUT             ; 表示超时
                      00262
0010                  00263 TMR0Int1
0010   110B           00264         bcf     INTCON,T0IF                     ; 清零溢出标志
0011   192F           00265         btfsc   COMFLAGS,FRSDATA                ; 检查是否处于接收模式
0012   2881           00266         goto    GetData
                      00267
0013   19AF           00268         btfsc   COMFLAGS,FTXDATA                ; 检查是否处于发送模式
0014   28BE           00269         goto    PutData
                      00270
0015                  00271 ExtInt
0015   1E0B           00272         btfss   INTCON,INTE                     ; 检查是否允许外部中断
0016   28F8           00273         goto    IntrEnd                         ; 以及是否可能发生中断
0017   1C8B           00274         btfss   INTCON,INTF                     ;
0018   28F8           00275         goto    IntrEnd
                      00276
0019   3026           00277         movlw   0x26                            ; +12 周期
001A   052F           00278         andwf   COMFLAGS,W                      ; 如果下一个字节是 ID 或起始位
001B   1D03           00279         btfss   STATUS,Z                        ; 的下降沿，则初始化
001C   286D           00280         goto    GetDataInit                     ; 从总线读一个字节
                      00281
001D   182F           00282         btfsc   COMFLAGS,FSYNCHBREAK            ; 是同步间隔？
001E   2832           00283         goto    CountSynchByte                  ; 是：则执行测量
                      00284                                                 ; 同步间隔字符的序列
001F   1683           00285         bsf     STATUS,RP0
消息 [302]：操作数中的寄存器不在 bank 0 中。确保 bank 位是正确的。
0020   1B01           00286         btfsc   OPTION_REG,INTEDG               ; 如果选择了上升沿，
0021   2827           00287         goto    CopySyncBreakLength             ; 则保存同步间隔的计数值
消息 [302]：操作数中的寄存器不在 bank 0 中。确保 bank 位是正确的。
0022   1701           00288         bsf     OPTION_REG,INTEDG               ; 否则设置上升沿敏感
0023   1283           00289         bcf     STATUS,RP0
                      00290
0024   0181           00291         clrf    TMR0                            ; 初始化所使用的计数器
0025   01A2           00292         clrf    HICOUNT                         ; 以测量同步间隔
0026   28F8           00293         goto    IntrEnd                         ; 并等待上升沿中断
                      00294
0027                  00295 CopySyncBreakLength
消息 [302]：操作数中的寄存器不在 bank 0 中。确保 bank 位是正确的。
0027   1301           00296         bcf     OPTION_REG,INTEDG               ; 设置下降沿敏感
```

```
0028   1283           00297         bcf     STATUS,RP0                      ; 以检测标识字节
0029   0801           00298         movf    TMR0,W                          ; 的起始位
002A   00A7           00299         movwf   SYNCLENGTH                      ; 保存计数器值
002B   0822           00300         movf    HICOUNT,W
002C   00A8           00301         movwf   SYNCLENGTH+1                    ; 如果高字节被清零
002D   1903           00302         btfsc   STATUS,Z                        ; 则所接收的值非同步间隔序列
002E   2868           00303         goto    LowerSynchLength                ; 因此启动另一个
                      00304                                                 ; 循环以检测同步间隔
002F   01A4           00305         clrf    COUNTEDGES                      ; 否则初始化同步
0030   142F           00306         bsf     COMFLAGS,FSYNCHBREAK            ; 字节的检测和测量
0031   28F8           00307         goto    IntrEnd
                      00308
                      00309 ; ---- CountSynchByte -----------------------------------------------------------
                      00310 ;
                      00311 ;       对五个下降沿之间的周期进行计数来计算单个位长度
                      00312 ;       用于同主节点通信。在第一个下降沿对计数寄存器清零。
                      00313 ;       在第五个下降沿之后，通过将计数值除以 8 并进行取整后可计算出位长度。
                      00314 ;
                      00315 ;
0032                  00316 CountSynchByte                                  ; 需要 17 个周期才能完成
0032   0824           00317         movf    COUNTEDGES,W
0033   1D03           00318         btfss   STATUS,Z                        ; 如果是第一个下降沿则
0034   2837           00319         goto    CountSynchEdges
0035   0181           00320         clrf    TMR0                            ;初始化字节长度计数器
0036   01A2           00321         clrf    HICOUNT
                      00322
0037                  00323 CountSynchEdges
0037   1924           00324         btfsc   COUNTEDGES,2                    ; 否则检查已经过边沿的计数值
0038   283B           00325         goto    GetSynchLength
0039   0AA4           00326         incf    COUNTEDGES,F                    ; 等待下一个下降沿
003A   28F8           00327         goto    IntrEnd
003B                  00328 GetSynchLength
003B   190B           00329         btfsc   INTCON,T0IF                     ; 如果计入了所有边沿
003C   0AA2           00330         incf    HICOUNT,F                       ; 则可通过
003D   0801           00331         movf    TMR0,W                          ; 将 16 位结果除以 8
003E   00AA           00332         movwf   BITLENGTH
003F   1003           00333         bcf     STATUS,C                        ; 计算实际位长度 （以周期计数为单位）
0040   0CA2           00334         rrf     HICOUNT,F
0041   0CAA           00335         rrf     BITLENGTH,F
0042   1003           00336         bcf     STATUS,C
0043   0CA2           00337         rrf     HICOUNT,F
0044   0CAA           00338         rrf     BITLENGTH,F
0045   1003           00339         bcf     STATUS,C
0046   0CA2           00340         rrf     HICOUNT,F
0047   0CAA           00341         rrf     BITLENGTH,F                     ; 结果保存在 BITLENGTH 中
0048   082A           00342         movf    BITLENGTH,W
0049   3C31           00343         sublw   .49                             ; BITLENGTH 值低于最快的
```

```
004A   1803           00344         btfsc   STATUS,C                        ; 比特率
004B   2868           00345        goto    LowerSynchLength
                      00346
                      00347 ; ---- CheckSynchBreakLength ----------------------------------------------------
                      00348 ;
                      00349 ;       将计数得到的位长度乘以 10 以确定同步间隔是否长于正常的数据字节。
                      00350 ;       一个正常的数据字节包含一个显性的起始位、 8 个数据位和一个隐性停
                      00351 ;       止位，因此如果测量的同步间隔的低电平时间长于 10 个位长度时间，
                      00352 ;       则表明它是同步间隔。
                      00353 ;
                      00354
004C                  00355 CheckSynchBreakLength
004C   01C4           00356         clrf    DUMMY+1                         ; 否 -> 计算同步间隔长度
004D   082A           00357         movf    BITLENGTH,W                     ; 保存位长度
004E   00C3           00358         movwf   DUMMY
004F   1003           00359         bcf     STATUS,C
0050   0DC3           00360         rlf     DUMMY,F                         ; 将位长度乘以 8
0051   0DC4           00361         rlf     DUMMY+1,F
0052   0DC3           00362         rlf     DUMMY,F
0053   0DC4           00363         rlf     DUMMY+1,F
0054   0DC3           00364         rlf     DUMMY,F
0055   0DC4           00365         rlf     DUMMY+1,F
0056   07C3           00366         addwf   DUMMY,F                         ; 并加上两倍的位长度值
0057   1803           00367         btfsc   STATUS,C                        ; 以实现乘以 10 的效果
0058   0AC4           00368         incf    DUMMY+1,F
0059   07C3           00369         addwf   DUMMY,F
005A   1803           00370         btfsc   STATUS,C
005B   0AC4           00371         incf    DUMMY+1,F
005C   0844           00372         movf    DUMMY+1,W
005D   0228           00373         subwf   SYNCLENGTH+1,W                  ; 首先检查高字节
005E   1C03           00374         btfss   STATUS,C                        ; 位长度 *10 < 同步长度
005F   2868           00375         goto    LowerSynchLength                ; 是 -> 未检测到报文头
0060   1D03           00376         btfss   STATUS,Z                        ; 位长度 *10 = 同步长度
0061   286B           00377         goto    HigherSyncLength                ; 否 -> 检测到报文头
0062   0843           00378         movf    DUMMY,W                         ; 高字节是相等的
0063   0227           00379         subwf   SYNCLENGTH,W                    ; 检查低字节
0064   1C03           00380         btfss   STATUS,C
0065   2868           00381         goto    LowerSynchLength
0066   1D03           00382         btfss   STATUS,Z
0067   286B           00383         goto    HigherSyncLength
                      00384
0068                  00385 LowerSynchLength
0068   01AF           00386         clrf    COMFLAGS                        ; 复位 lin 从节点
0069   160B           00387         bsf     INTCON,INTE                     ; 允许外部中断
006A   28F8           00388         goto    IntrEnd                         ; 以产生第一个下降沿
                      00389
006B                  00390 HigherSyncLength
```

```
006B   14AF          00391         bsf     COMFLAGS,FID                    ; 输入的下一个数据字节为
006C   28F8          00392         goto    IntrEnd                         ; 标识字节
                     00393
                     00394 ; ---- GetDataInit ------------------------------------------------------------
                     00395 ;
                     00396 ;       当检测到起始位的下降沿时，这一函数对位长度定时器 （TMR0）进行初始化以在位时间
                     00397 ;       的中间时刻读该位状态。
                     00398 ;
                     00399 ;
                     00400 ;       对于典型通信速度为 9600 至 19.2k 波特率的情况，我们可将 timer 0 设置为
                     00401 ;       1.5 个位时间，以使得跳过起始位的剩余时间并通过第一个数据位进行中途唤醒。
                     00402 ;       然而对于较慢的波特率，如低于 6k 波特率的情况， 1.5 个位时间可能导致 8 位
                     00403 ;       定时器溢出。在这种情况下，可将 timer 0 设置为半个位时间这样即可通过起始
                     00404 ;       位实现中途唤醒并对位计数器进行调整以对起始位的剩余时间进行计数。
                     00405 ;
                     00406 ;
                     00407 ;
                     00408
006D                 00409 GetDataInit
006D   01B1          00410         clrf    COMBUFFER                       ; 清零输入缓冲区
006E   01A9          00411         clrf    BITNBR                          ; 和位计数器
006F   132F          00412         bcf     COMFLAGS,FCOMPLETE              ; 清零通信结束
                     00413                                                 ; 指示
0070   082A          00414         movf    BITLENGTH,W                     ; 检查波特率是否小于
0071   3CA6          00415         sublw   .166                            ; 6 k
0072   1C03          00416         btfss   STATUS,C                        ; 如果高于，则
0073   2879          00417         goto    SetHalfStartbitLength           ; 将中间位置设置在起始位中
                     00418
0074   1003          00419         bcf     STATUS,C                        ; 否则通过计算一个半位长度的时间
0075   0AA9          00420         incf    BITNBR,F                        ; 将中间位置设置在
0076   0C2A          00421         rrf     BITLENGTH,W                     ; 第一个数据位中
0077   072A          00422         addwf   BITLENGTH,W
0078   287A          00423         goto    GetDataInitEnd
                     00424
0079                 00425 SetHalfStartbitLength
0079   0C2A          00426         rrf     BITLENGTH,W                     ; 中间停止位
                     00427
007A                 00428 GetDataInitEnd
007A   3AFF          00429         xorlw   0xff                            ; 对定时器按位取反
007B   3E3E          00430         addlw   (.62)                           ; 校正定时器以将中间位置设置在停止位中
007C   0081          00431         movwf   TMR0
007D   3020          00432         movlw   0x20                            ; INTE 设为禁止， T0IE 设为允许，清零标志
007E   008B          00433         movwf   INTCON                          ; 设置中断
007F   152F          00434         bsf     COMFLAGS,FRSDATA                ; 指示处于接收模式
0080   28F8          00435         goto    IntrEnd
                     00436
                     00437 ; ---- GetData ----------------------------------------------------------------
```

```
                        00438 ;
                        00439 ;       Timer 0 已被设置为在此返回从而在此位时间的中间时刻对接收到的位进行测试。
                        00440 ;       该函数对位状态进行采样并将该位的值循环移入数据缓冲区。
                        00441 ;
                        00442 ;
                        00443 ;       如果这是个标识字节，将调用 DecodeIDTable 以判别下一步做什么 （接收报文、
                        00444 ;       发送报文或是忽略报文）。
                        00445 ;       如果是一个数据字节，该字节将被存放在前面调用 DecodeIDTable 建立的缓冲区中。
                        00446 ;       接收到所有数据后，将计算校验和并将其与发送值进行比较。如果没有错误，
                        00447 ;       FCOMPLETE 标志将被置 1 以表示应对缓冲区中的数据进行处理。
                        00448 ;
                        00449 ;
                        00450 ;
                        00451 ;       在 TMR0 溢出之后进行采样，子程序将把引脚值复制到通信缓冲区并对停止位电平进行
                        00452 ;       检查以查看数据是否正确。在接收到字节的所有位后，会有以下两种可能性。
                        00453 ;
                        00454 ;
                        00455 ;       1.      接收到的字节是标识字节。此刻，子程序对该字节进行分析，根据函数
                        00456 ;               “DecodeIDTable”的指示来建立接收缓冲区、发送缓冲区或什么都不做。
                        00457 ;
                        00458 ;
                        00459 ;
                        00460 ;       2.      接收到的字节是响应帧的数据字节。
                        00461 ;               数据存放在接收缓冲区中，该值加上先前的进位位一起构成模 256 校验和。
                        00462 ;               最后一个字节是报文校验和，该值将与计算的校验和进行比较以检查报文的
                        00463 ;               完整性。
                        00464 ;
                        00465 ;
                        00466
0081                    00467 GetData                                   ; 延迟 13 个周期
0081   01A2             00468         clrf    TIMEOUTLO                 ; 复位超时计数器
0082   01A3             00469        clrf    TIMEOUTHI
0083                    00470 CheckBitPosition
0083   3009             00471         movlw   .9                        ; 还剩余几位
0084   0629             00472         xorwf   BITNBR,W                  ; 接收到停止位？
0085   1903             00473         btfsc   STATUS,Z                  ; 否：备份接收引脚值
0086   2891             00474        goto    GetStopbit
                        00475
0087   1C06             00476         btfss   PORTB,RSLINEPIN           ; 使用进位标志在数据缓冲区中
0088   1003             00477         bcf     STATUS,C                  ; 备份端口值
0089   1806             00478        btfsc   PORTB,RSLINEPIN
008A   1403             00479        bsf     STATUS,C
008B   0CB1             00480         rrf     COMBUFFER,F               ; 将进位位移入缓冲区
                        00481
008C                    00482 GetDataSetTMR
008C   0AA9             00483         incf    BITNBR,F                  ; 计数接收到的位
008D   092A             00484         comf    BITLENGTH,W               ; 并用正确的值
```

```
008E   3E1F          00485         addlw   (.31)                          ; 将 TMR0 中间对齐下一个位
008F   0081          00486         movwf   TMR0
0090   28BD          00487         goto    GetDataEnd
                     00488
0091                 00489 GetStopbit
0091   1C06          00490         btfss   PORTB,RSLINEPIN                ; 检查停止位的极性
0092   15AE          00491         bsf     ERRORFLAGS,FBITERROR           ; 如果为低电平则置 1 错误标志
                     00492
0093   18AF          00493         btfsc   COMFLAGS,FID                   ; 如果接收到标识字节
0094   28AF          00494         goto    GetAction                      ; 对该字节进行检查以初始化
                     00495                                                ; 要求的从模式
0095   032B          00496         decf    DATABLOCKLENGTH,W              ; 否则，如果接收了所有字节
0096   1903          00497         btfsc   STATUS,Z
0097   28A7          00498         goto    GetCheckCRC                    ; 那么最后一个字节应是校验和
0098   03AB          00499         decf    DATABLOCKLENGTH,F              ; 否则等待下一个字节
                     00500
0099   082E          00501         movf    ERRORFLAGS,W                   ; 检查是否有错误
009A   1D03          00502         btfss   STATUS,Z                       ; 如果检测到错误
009B   28A5          00503         goto    SetNextLoc                     ; 不保存该值
                     00504
009C   1AAF          00505         btfsc   COMFLAGS,FLISTENONLY           ; 或如果从节点监视线路
009D   28A5          00506         goto    SetNextLoc                     ; 忽略下一步骤
                     00507
009E   0830          00508         movf    BUFFERPTR,W                    ; 获取指针
009F   0084          00509         movwf   FSR                            ; 并指向地址单元
00A0   0831          00510         movf    COMBUFFER,W                    ; 来获取实际数据
00A1   0080          00511         movwf   INDF                           ; 并将其传输到数据块
                     00512
                     00513 ;
                     00514 ;       对接收到的数据字节采用模 256 进行校验和的计算
                     00515 ;       DATACRC = DATACRC + COMBUFFER + Carry
                     00516 ;
00A2   07B2          00517         addwf   DATACRC,F                      ; 将新的数据加到 CRC
00A3   1803          00518         btfsc   STATUS,C                       ; 如果 mod 256 产生溢出，
00A4   0AB2          00519         incf    DATACRC,F                      ; 则加上进位位
                     00520
00A5                 00521 SetNextLoc
00A5   0AB0          00522         incf    BUFFERPTR,F                    ; 指向下一个地址单元
00A6   28B0          00523         goto    InitGetData
                     00524
00A7                 00525 GetCheckCRC
00A7   1AAF          00526         btfsc   COMFLAGS,FLISTENONLY           ; 如果从节点只读入报文，
00A8   28B7          00527         goto    GetDataFinish                  ; 则忽略校验和计算
                     00528
                     00529 ;
                     00530 ; 对数据块取 256 模再取反加上 CRC 所产生的和等于 0xFF
                     00531 ; 计算 0xFF - SUM[RSDATAFIELD] = 接收到的 CRC
```

```
                        00532
00A9   30FF             00533          movlw   0xFF                            ; 对 CRC 按位取反
00AA   0632             00534          xorwf   DATACRC,W
                        00535
00AB   0631             00536          xorwf   COMBUFFER,W                     ; 检查接收到的 CRC 与
00AC   1D03             00537          btfss   STATUS,Z                        ; 计算得到的 CRC
00AD   152E             00538          bsf     ERRORFLAGS,FCRCERROR            ; 如果不相同，置 1 标志
00AE   28B7             00539          goto    GetDataFinish
                        00540
00AF                    00541 GetAction
00AF   212C             00542          call    CheckIdentifierByte             ; 解码并检查 ID 字节
                        00543
00B0                    00544 InitGetData
00B0   01A9             00545          clrf    BITNBR                          ; 清零寄存器以准备下一个操作
00B1   1DAF             00546          btfss   COMFLAGS,FTXDATA                ; 如果下一个操作是发送
00B2   28BA             00547          goto    GetDataFinish+3                 ; 仅对中断进行初始化
                        00548
00B3   120B             00549          bcf     INTCON,INTE                     ; 禁止外部中断
00B4   092A             00550          comf    BITLENGTH,W                     ; 在发送数据之前进行延迟
00B5   0081             00551         movwf   TMR0
00B6   28BD             00552         goto    GetDataEnd
                        00553
00B7                    00554 GetDataFinish
00B7   30C0             00555          movlw   0xC0
00B8   05AF             00556          andwf   COMFLAGS,F                      ; 清零所有标志位
00B9   172F             00557          bsf     COMFLAGS,FCOMPLETE              ; 指示接收模式结束
00BA   3010             00558          movlw   0x10                            ; T0IE 设为 0， INTE 设为 1，清零标志
00BB   008B             00559          movwf   INTCON                          ; 等待下一个同步间隔下降沿
00BC   0181             00560          clrf    TMR0
                        00561
00BD                    00562 GetDataEnd
                        00563 ;        bcf     INTCON,T0IF
00BD   28F8             00564          goto    IntrEnd
                        00565
                        00566 ; ---- PutData ---------------------------------------------------------------------
                        00567 ;
                        00568 ;        这一子程序仅移出所有需要的、包括起始位和停止位在内的数据位
                        00569 ;        通过置 1 停止位，子程序将计算模 256 校验和
                        00570 ;        并将在所有数据位发送完之后发送校验和字节。
                        00571 ;
                        00572
00BE                    00573 PutData
00BE   01A2             00574         clrf    TIMEOUTLO
00BF   01A3             00575         clrf    TIMEOUTHI
                        00576
00C0   0829             00577        movf    BITNBR,W                         ; 如果首次调用置 1 起始位
00C1   1003             00578         bcf     STATUS,C
```

```
00C2   1903          00579        btfsc   STATUS,Z
00C3   28D7          00580        goto    SetStartbit
                     00581
00C4   0829          000582        movf    BITNBR,W                    ; 如果移出所有位
00C5   3A09          000583        xorlw   .9                          ; 则发送停止位
00C6   1903          00584        btfsc   STATUS,Z
00C7   28E3          00585        goto    SetStopbit
                     00586
00C8                 00587 ShiftDatabitOut
00C8   0CB1          00588         rrf     COMBUFFER,F                 ; 否则将位保存到 C 中
00C9   0806          00589         movf    PORTB,W                     ; 获取端口值
00CA   39EF          00590         andlw   ((1<<TXLINEPIN)^0xFF)       ; 清零使用过的位
00CB   1803          00591         btfsc   STATUS,C                    ; 随后设置发送线值
00CC   3810          00592         iorlw   (1<<TXLINEPIN)              ; 如果必要的话
00CD   0086          00593        movwf   PORTB
                     00594
00CE                 00595 CheckBitPending
00CE   0DC3          00596         rlf     DUMMY,F                     ; 将进位位移至变量中
00CF   0606          00597         xorwf   PORTB,W                     ; 测试输入位的极性
00D0   3901          00598         andlw   (1<<RSLINEPIN)              ; 屏蔽所有指示位
00D1   1D03          00599         btfss   STATUS,Z                    ; 如果极性相同，则正常
00D2   15AE          00600         bsf     ERRORFLAGS,FBITERROR
                     00601
00D3                 00602 PutDataSetTMR0
00D3   092A          00603         comf    BITLENGTH,W                 ; 预装载位长度值到定时器中
00D4   3E27          00604         addlw   (.39)                       ; 校正定时器
00D5   0081          00605         movwf   TMR0
00D6   28F7          00606        goto    PutDataEnd
                     00607
00D7                 00608 SetStartbit
00D7   1206          00609        bcf     PORTB,TXLINEPIN
00D8   0830          00610        movf    BUFFERPTR,W
00D9   0084          00611         movwf   FSR                         ; 并指向它
00DA   0800          00612         movf    INDF,W                      ; 获取实际数据
00DB   00B1          00613        movwf   COMBUFFER
                     00614
                     00615 ;
                     00616 ; 使用模 256 产生 CRC  -> DATACRC = COMBUFFER + DATACRC + 进位位
                     00617 ;
00DC                 00618 CalculateTxCRC
00DC   07B2          00619         addwf   DATACRC,F                   ; 生成校验和
00DD   1803          00620         btfsc   STATUS,C                    ; 将进位位加到 data_CRC 中
00DE   0AB2          00621         incf    DATACRC,F                   ; 如果发生溢出
00DF   092A          00622         comf    BITLENGTH,W                 ; 预装载位长度至定时器
00E0   3E2E          00623         addlw   (.46)                       ; 校正定时器
00E1   0081          00624        movwf   TMR0
00E2   28F7          00625        goto    PutDataEnd
```

```
                    00626
00E3                00627 SetStopbit
00E3   30FF         00628         movlw   -1
00E4   00A9         00629         movwf   BITNBR                          ; 准备清零寄存器
00E5   1606         00630         bsf     PORTB,TXLINEPIN                 ; 设置发送线状态为停止位
00E6   0181         00631         clrf    TMR0                            ; 等待 1*Tbit + x us
00E7   0AB0         00632         incf    BUFFERPTR,F                     ; 数组指向下一个单元
00E8   03AB         00633         decf    DATABLOCKLENGTH,F               ; 发送所有数据字节吗？
00E9   082B         00634         movf    DATABLOCKLENGTH,W
00EA   1903         00635         btfsc   STATUS,Z                        ; 移出包括 CRC 字节在内
00EB   28F4         00636         goto    PutDataFinish                   ; 的所有数据字节？
00EC   3A01         00637         xorlw   .1                              ; 否：仅是 CRC 字节？
00ED   1D03         00638         btfss   STATUS,Z
00EE   28F7         00639         goto    PutDataEnd                      ; 否：那么等待下一个数据位
                    00640
                    00641 ;
                    00642 ; 产生取反的模 256 校验和
                    00643 ; COMBUFFER = DATACRC XOR 0xFF，所以 SUM(TxData_field)+DATACRC = 0xFF
                    00644 ;
                    00645
00EF                00646 SetPtr2CRC
00EF   30FF         00647         movlw   0xFF
00F0   06B2         00648         xorwf   DATACRC,F                       ; 产生正确的校验和
                    00649
00F1   3032         00650         movlw   DATACRC                         ; 获得校验和寄存器地址
00F2   00B0         00651         movwf   BUFFERPTR                       ; 并保存该值
00F3   28F7         00652         goto    PutDataEnd
                    00653
00F4                00654 PutDataFinish
00F4   01AF         00655         clrf    COMFLAGS                        ; 复位通信标志
00F5   172F         00656         bsf     COMFLAGS,FCOMPLETE              ; 指示完成标志
00F6   160B         00657         bsf     INTCON,INTE                     ; 和具有同步间隔检测
                    00658                                                 ; 的新通信开始
00F7                00659 PutDataEnd
00F7   0AA9         00660         incf    BITNBR,F                        ; 将位计数器设置为零
                    00661
                    00662 ; ---- IntrEnd ------------------------------------------------------------------
                    00663 ;
                    00664 ;       恢复备份寄存器并清零中断标志，从中断返回。
                    00665 ;
                    00666
00F8                00667 IntrEnd
00F8   30F0         00668         movlw   0xF0                            ; 清零所有等待处理的中断标志
00F9   058B         00669         andwf   INTCON,F
00FA   168B         00670         bsf     INTCON,T0IE                     ; 并置 1 timer0 溢出中断标志
                    00671
00FB   0821         00672         movf    COPYSTATUS,W                    ; 恢复状态寄存器
```

```
00FC   0083           00673         movwf   STATUS
00FD   0EA0           00674         swapf   COPYWREG,F                      ; 且从下一个 ram 存储区获取
00FE   0E20           00675         swapf   COPYWREG,W                      ; 旧的 W 值
00FF   0009           00676         retfie
                      00677
                      00678
                      00679 ; #### 子程序 ################################################################
                      00680 ;
                      00681 ;
                      00682 ; #### init_LinSlave ########################################################
                      00683 ;
                      00684 ;       清零 lin 从节点使用的所有 ram 单元并对端口引脚和中断标志进行初始化
                      00685 ;
                      00686 ;
                      00687
0100                  00688 InitLinSlave
0100   3020           00689         movlw   LINSLAVERAM
0101   0084           00690         movwf   FSR
0102   3024           00691         movlw   LINSLAVEBLOCKLENGTH
0103                  00692 ClearLINusedRAM
0103   0180           00693         clrf    INDF                            ; 清零寄存器
0104   0A84           00694         incf    FSR,F                           ; 指向下一个单元
0105   0804           00695         movf    FSR,W
0106   3C44           00696         sublw   (LINSLAVEBLOCKLENGTH+LINSLAVERAM)
0107   1D03           00697         btfss   STATUS,Z                        ; 全部清零了吗？
0108   2903           00698         goto    ClearLINusedRAM                 ; 否：下一次循环
                      00699
0109   1606           00700         bsf     PORTB,TXLINEPIN                 ; 编程开始时端口引脚处于高电平
010A   1683           00701         bsf     STATUS,RP0
010B   1406           00702         bsf     PORTB,RSLINEPIN                 ; 接收线作为输入
010C   1206           00703         bcf     PORTB,TXLINEPIN                 ; 而发送线为输出
010D   3008           00704         movlw   0x08                            ; TMR0 设置为计数器驱动模式
消息 [302]：操作数中的寄存器不在 bank 0 中。确保 bank 位是正确的。
010E   0081           00705         movwf   OPTION_REG                      ; 带内部时钟
消息 [302]：操作数中的寄存器不在 bank 0 中。确保 bank 位是正确的。
010F   1301           00706         bcf     OPTION_REG,INTEDG
0110   1283           00707         bcf     STATUS,RP0
0111   108B           00708         bcf     INTCON,INTF
0112   160B           00709         bsf     INTCON,INTE
0113   178B           00710         bsf     INTCON,GIE
0114   0008           00711         return
                      00712
                      00713 ; #### InitWakeupLIN ########################################################
                      00714 ;
                      00715 ;       通过将总线拉为低电平持续 7 个位时间，并在此之后跟随一个高电平停止位来唤醒总线。
                      00716 ;       用在总线进入休眠之后，且节点需要唤醒总线的情况下。一旦总线被重新唤醒，主节点将开始查询从节点
                      00717 ;       状态以查找唤醒原因。
```

```
                    00718 ;
                    00719 ;
                    00720
0115                00721 InitWakeupLIN
0115   082A         00722         movf    BITLENGTH,W
0116   1903         00723         btfsc   STATUS,Z                        ; 如果无需再进行通信，
0117   2926         00724         goto    InitWakeupEnd                   ; 则无须执行该函数
0118   3001         00725         movlw   .1
0119   00AB         00726         movwf   DATABLOCKLENGTH                 ; 使用 CRC 寄存器作为数据容器
011A   3080         00727         movlw   0x80                            ; 获得一个字节数据块
011B   00B2         00728         movwf   DATACRC
011C   3032         00729         movlw   DATACRC                         ; 初始化数据指针指向 CRC
011D   00B0         00730         movwf   BUFFERPTR
011E   01A9         00731         clrf    BITNBR                          ; 并开始发送操作
011F   15AF         00732         bsf     COMFLAGS,FTXDATA
                    00733
0120   0181         00734         clrf    TMR0                            ; 在发送唤醒序列之前
                    00735                                                 ; 执行一个延迟
0121   30A0         00736         movlw   0xA0                            ; 设置 TMR0 中断
0122   008B         00737         movwf   INTCON
0123   19AF         00738         btfsc   COMFLAGS,FTXDATA
0124   2923         00739         goto    $-1
0125   13AF         00740         bcf     COMFLAGS,FSLEEPMODE             ; 等待唤醒序列结束
0126                00741 InitWakeupEnd
0126   0008         00742         return
                    00743
                    00744 ; #### CheckIdentifierByte ########################################################
                    00745 ;
                    00746 ;       在检测到同步间隔和同步字节并计算出位长度后，由中断调用该函数。
                    00747 ;
                    00748 ;       从总线读取标识字节并在此刻对字节进行检查以判断奇偶校验位是否正确，随后提取块长度和 ID 值，
                    00749 ;       不管 ID 为何值，都将设置处理程序模式。
                    00750 ;
                    00751 ;
                    00752 ;
0127                00753 DataBlockLengthTable                            ; 用于接收操作的解码表
0127   0782         00754         addwf   PCL,F                           ; 字节长度
0128   3403 3403 3405 00755        DT      3,3,5,9                         ; 对于校验和还有一个字节
       3409
                    00756                                                 ; 字节位于数据数组的末尾
                    00757
012C                00758 CheckIdentifierByte
012C   01AF         00759         clrf    COMFLAGS                        ; 清零所有标志
012D   0831         00760         movf    COMBUFFER,W                     ; 获取标识字段
012E   3A80         00761         xorlw   0x80                            ; 并检测是否有休眠命令
012F   1903         00762         btfsc   STATUS,Z
0130   29A0         00763         goto    ActionBusSleep                  ; 是：暂停函数执行
```

```
                    00764
0131                00765 CheckEvenParityBit6                       ; 对 1 的个数进行计数以进行偶校验
0131   01A9         00766         clrf    BITNBR
0132   1831         00767         btfsc   COMBUFFER,0
0133   0AA9         00768         incf    BITNBR,F
0134   18B1         00769         btfsc   COMBUFFER,1
0135   0AA9         00770         incf    BITNBR,F
0136   1931         00771         btfsc   COMBUFFER,2
0137   0AA9         00772         incf    BITNBR,F
0138   1A31         00773         btfsc   COMBUFFER,4
0139   0AA9         00774         incf    BITNBR,F
                    00775
013A   1B31         00776         btfsc   COMBUFFER,6                ; 检查 bit6 的值是否为
013B   0AA9         00777         incf    BITNBR,F                  ; 偶数值
013C   1829         00778         btfsc   BITNBR,0                  ; 如果是偶数，随后检查奇校验位
013D   294A         00779         goto    SetParityError
013E                00780 CheckOddParityBit7                        ; 对 1 的个数进行计数以进行奇校验
013E   01A9         00781         clrf    BITNBR                    ; 并取反
013F   18B1         00782         btfsc   COMBUFFER,1
0140   0AA9         00783         incf    BITNBR,F
0141   19B1         00784         btfsc   COMBUFFER,3
0142   0AA9         00785         incf    BITNBR,F
0143   1A31         00786         btfsc   COMBUFFER,4
0144   0AA9         00787         incf    BITNBR,F
0145   1AB1         00788         btfsc   COMBUFFER,5
0146   0AA9         00789         incf    BITNBR,F
                    00790
0147   1FB1         00791         btfss   COMBUFFER,7
0148   0AA9         00792         incf    BITNBR,F                  ; 如果为奇，则拆分
0149   1829         00793         btfsc   BITNBR,0                  ; 数据块长度
014A                00794 SetParityError
014A   14AE         00795         bsf     ERRORFLAGS,FIDPARITYERROR ; 如果有错误，
                    00796                                           ; 置 1 奇偶校验错误位
014B                00797 GetPreID
014B   0831         00798         movf    COMBUFFER,W               ; 获取 id 值
014C   393F         00799         andlw   0x3F                      ; 以及块长度子集
014D   00AC         00800         movwf   PREIDNUMBER               ; 并保存该值
                    00801
014E                00802 DecodeBlockLength
014E   3001         00803         movlw   high DataBlockLengthTable ; 获取解码表的地址
014F   008A         00804         movwf   PCLATH
                    00805
0150   0E31         00806         swapf   COMBUFFER,W               ; 交换 bit3,4 到 bit0,1 的值
0151   3903         00807         andlw   .3                        ; 清零其他位
0152   2127         00808         call    DataBlockLengthTable      ; 且对结果进行解码
0153   00AB         00809         movwf   DATABLOCKLENGTH           ; 保存使用的块长度
                    00810                                           ; 包括校验和字节
```

```
0154   01B2           00811         clrf    DATACRC                         ; 对校验和以及错误标志寄存器
0155   01AE           00812         clrf    ERRORFLAGS                      ; 进行初始化
                      00813
0156                  00814 DecodeIdNumber
0156   162F           00815         bsf     COMFLAGS,FCOMDATA               ; 置 1 通信标志
                      00816
0157   3001           00817         movlw   HIGH DecodeIDTable              ; 使用表地址对高位程序计数器
0158   008A           00818         movwf   PCLATH                          ; 进行初始化
                      00819
0159   0831           00820         movf    COMBUFFER,W                     ; 获得缓冲区值
015A   390F           00821         andlw   .15
015B   00AD           00822         movwf   IDNUMBER                        ; 保存 ID 值
015C   00B1           00823         movwf   COMBUFFER                       ; 通过将 id 乘以 4
015D   0DB1           00824         rlf     COMBUFFER,F                     ; 产生跳转宽度
015E   0D31           00825         rlf     COMBUFFER,W                     ; 并利用 PCL
015F   0782           00826         addwf   PCL,F                           ; 进行跳转
                      00827
                      00828 ; ---- DecodeIDTable ------------------------------------------------------------
                      00829 ;
                      00830 ;       使用 ID 值解码表
                      00831 ;
                      00832 ;       此时用户可确定从节点如何根据解码后的标识值采取相应的操作。
                      00833 ;       还能够决定将接收数据存放到何处或从何处发送数据。
                      00834 ;
                      00835 ;
                      00836 ;       可能有三种模式：
                      00837 ;               MacroRsMode (ptr)       ->  从总线读取数据并将其保存到
                      00838 ;                                           给定缓冲区 (ptr) 中
                      00839 ;               MacroTxMode (ptr)       ->  将给定缓冲区中的数据发送到总线上。
                      00840 ;
                      00841 ;               MacroLstMode            ->  仅对总线进行监视
                      00842 ;
                      00843 ;       在监视模式中，用户可使用 PREIDNUMBER 的值处理 64 条命令而无须任何数据报文。
                      00844 ;
                      00845
0160                  00846 DecodeIDTable
                      00847         MacroLstMode
0160   16AF               M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                          M                                                 ; 无须保存接收到的数据
0161   0000               M         nop                                     ; 有必要获得一个正确的
0162   0000               M         nop                                     ; 跳转表长度
0163   3400               M         retlw   0
                      00848         MacroTxMode  ( BUTTONPRESSED1 )  ; ID1：发送按钮按下次数
0164   15AF               M         bsf     COMFLAGS,FTXDATA                ; 指示发送模式
0165   3033               M         movlw   ( BUTTONPRESSED1 )              ; 获得所需的 ram 地址单元
                          M                                                 ; 该地址存放将读取的数据
0166   00B0               M         movwf   BUFFERPTR                       ; 并对指针进行初始化
```

```
0167   3400              M         retlw   0
                       00849       MacroLstMode
0168   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
0169   0000              M         nop                                   ; 有必要获得一个
016A   0000              M         nop                                   ; 正确的跳转表长度
016B   3400              M         retlw   0
                       00850       MacroLstMode
016C   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
016D   0000              M         nop                                   ; 有必要获得一个
016E   0000              M         nop                                   ; 正确的跳转表长度
016F   3400             M          retlw   0
                       00851       MacroRsMode  ( COMPLED1 )             ; ID4：接收 LED 计数器
0170   152F              M         bsf     COMFLAGS,FRSDATA              ; 指示接收模式
0171   303B              M         movlw   ( COMPLED1 )                  ; 获取所需的 ram 地址单元
                         M                                               ; 用于存放数据
0172   00B0              M         movwf   BUFFERPTR                     ; 并对指针进行初始化
0173   3400              M         retlw   0
                       00852       MacroLstMode
0174   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
0175   0000              M         nop                                   ; 有必要获得一个
0176   0000              M         nop                                   ; 正确的跳转表长度
0177   3400              M         retlw   0
                       00853       MacroLstMode
0178   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
0179   0000              M         nop                                   ; 有必要获得一个
017A   0000              M         nop                                   ; 正确的跳转表长度
017B   3400              M         retlw   0
                       00854       MacroLstMode
017C   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
017D   0000              M         nop                                   ; 有必要获得一个
017E   0000              M         nop                                   ; 正确的跳转表长度
017F   3400              M         retlw   0
                       00855       MacroLstMode
0180   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
0181   0000              M         nop                                   ; 有必要获得一个
0182   0000              M         nop                                   ; 正确的跳转表长度
0183   3400              M         retlw   0
                       00856       MacroLstMode
0184   16AF              M         bsf     COMFLAGS,FLISTENONLY          ; 指示接收模式
                         M                                               ; 无须保存接收到的数据
0185   0000              M         nop                                   ; 有必要获得一个
```

```
0186   0000              M         nop                                     ; 正确的跳转表长度
0187   3400             M          retlw   0
                      00857        MacroLstMode
0188   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
0189   0000              M         nop                                     ; 有必要获得一个
018A   0000              M         nop                                     ; 正确的跳转表长度
018B   3400             M          retlw   0
                      00858        MacroLstMode
018C   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
018D   0000              M         nop                                     ; 有必要获得一个
018E   0000              M         nop                                     ; 正确的跳转表长度
018F   3400              M         retlw   0
                      00859        MacroLstMode
0190   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
0191   0000              M         nop                                     ; 有必要获得一个
0192   0000              M         nop                                     ; 正确的跳转表长度
0193   3400              M         retlw   0
                      00860        MacroLstMode
0194   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
0195   0000              M         nop                                     ; 有必要获得一个
0196   0000              M         nop                                     ; 正确的跳转表长度
0197   3400              M         retlw   0
                      00861        MacroLstMode
0198   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
0199   0000              M         nop                                     ; 有必要获得一个
019A   0000              M         nop                                     ; 正确的跳转表长度
019B   3400              M         retlw   0
                      00862        MacroLstMode
019C   16AF              M         bsf     COMFLAGS,FLISTENONLY            ; 指示接收模式
                         M                                                 ; 无须保存接收到的数据
019D   0000              M         nop                                     ; 有必要获得一个
019E   0000              M         nop                                     ; 正确的跳转表长度
019F   3400              M         retlw   0
                      00863
                      00864
                      00865 ; ---- ActionBusSleep -----------------------------------------------------------
                      00866 ;
                      00867 ;       一旦遇到这一序列，主节点将不再同从节点进行通信因此它将发命令以使总线挂起。
                      00868 ;       在总线读操作完成之后，主任务将能够激活唤醒序列以向主节点发送相应的命令来重新启动通信。
                      00869 ;
                      00870 ;
                      00871 ;
```

```
                    00872
01A0                00873 ActionBusSleep
01A0    17AF        00874         bsf     COMFLAGS,FSLEEPMODE
01A1    3003        00875         movlw   .3
01A2    00AB        00876         movwf   DATABLOCKLENGTH                 ; 分解长度控制位
01A3    3080        00877         movlw   0x80
01A4    00AD        00878         movwf   IDNUMBER                        ; 休眠时，设置标识 id
                    00879         MacroRsMode ( RSDATAFIELD )             ; 保存接收到的两个字节
01A5    152F            M         bsf     COMFLAGS,FRSDATA                ; 指示接收模式
01A6    303B            M         movlw   ( RSDATAFIELD )                 ; 获得所需的 ram 地址单元
                        M                                                 ; 该单元用于存放数据
01A7    00B0            M         movwf   BUFFERPTR                       ; 并初始化指针
01A8    3400            M         retlw   0
                    00880
                    00881 ; #### 主程序 ##########################################################################
                    00882
01A9                00883 Main
                    00884
01A9    300A        00885         movlw   .10
01AA    00C5        00886         movwf   COMPERATOR                      ; 在 x 次计数之后点亮 LED1 和 LED2
01AB    00C8        00887         movwf   DEBOUNCECOUNTER
                    00888
01AC    01CA        00889         clrf    EDGEDETECT
                    00890
01AD    1683        00891         bsf     STATUS,RP0
01AE    301F        00892         movlw   0x1F                            ; 将 PORTB7..5 设置为输出
01AF    0086        00893         movwf   PORTB                           ; 以控制 LED
01B0    1283        00894         bcf     STATUS,RP0
                    00895
01B1    2100        00896         call    InitLinSlave                    ; 初始化 lin 总线函数
                    00897
01B2                00898 MainLoop
01B2    1F2F        00899         btfss   COMFLAGS,FCOMPLETE              ; 报文帧结束？
01B3    29B7        00900         goto    CheckLevel                      ; 否 -> 执行其余的主任务
                    00901
01B4    132F        00902         bcf     COMFLAGS,FCOMPLETE              ; 清除标志
01B5    3080        00903         movlw   (1<<LED3)                       ; 并翻转 LED 状态
01B6    0686        00904         xorwf   PORTB,F
                    00905
01B7                00906 CheckLevel
01B7    0845        00907         movf    COMPERATOR,W                    ; 将第一个值与限定值
01B8    023B        00908         subwf   COMPLED1,W                      ; 进行比较
01B9    1C03        00909         btfss   STATUS,C                        ; 如果不相等
01BA    29BE        00910         goto    CheckNextLevel                  ; 检查下一个值
                    00911
01BB    01BB        00912         clrf    COMPLED1                        ; 否则处理该值
01BC    3020        00913         movlw   (1<<LED1)                       ; 并翻转 LED1 状态
```

```
01BD   0686          00914         xorwf   PORTB,F
                     00915
01BE                 00916 CheckNextLevel
01BE   0845          00917         movf    COMPERATOR,W                    ; 比较第二个值
01BF   023C          00918         subwf   COMPLED2,W
01C0   1C03          00919         btfss   STATUS,C
01C1   29C5          00920         goto    TestButtons
                     00921
01C2   01BC          00922         clrf    COMPLED2                        ; 如果相等
01C3   3040          00923         movlw   (1<<LED2)                       ; 则翻转 LED2 状态
01C4   0686          00924         xorwf   PORTB,F
                     00925
01C5                 00926 TestButtons
01C5   0BC8          00927         decfsz  DEBOUNCECOUNTER,F
01C6   29B2          00928         goto    MainLoop                        ; 执行循环
                     00929
                     00930
01C7   3006          00931         movlw   (1<<BUTTON1) | (1<<BUTTON2)     ; 屏蔽所使用的
01C8   0506          00932         andwf   PORTB,W                         ; 端口引脚
01C9   00C9          00933         movwf   COPYPORTB                       ; 并保存结果
                     00934
01CA                 00935 CheckEdgeChange
01CA   064A          00936         xorwf   EDGEDETECT,W                    ; 检查实际值和最后一个值以检测
01CB   0549          00937         andwf   COPYPORTB,W                     ; 输入端是否出现任何上升沿
01CC   1903          00938          btfsc   STATUS,Z                        ;
01CD   29D7          00939         goto    ButtomTestEnd                   ; 没有 -> 则执行循环
                     00940
01CE   00C8          00941         movwf   DEBOUNCECOUNTER                 ; 保存改变状态的引脚值
                     00942
01CF                 00943 RisingEdgeButtom1
01CF   3002          00944         movlw   (1<<BUTTON1)                    ; 屏蔽按钮 1
01D0   0548          00945         andwf   DEBOUNCECOUNTER,W               ; 按钮 1 是否按下？
01D1   1D03          00946         btfss   STATUS,Z
01D2   0AB3          00947         incf    BUTTONPRESSED1,F                ; 是：计数上升沿
                     00948
01D3                 00949 RisingEdgeButtom2
01D3   3004          00950         movlw   (1<<BUTTON2)                    ; 屏蔽按钮 2
01D4   0548          00951         andwf   DEBOUNCECOUNTER,W               ; 按钮 2 是否按下？
01D5   1D03          00952         btfss   STATUS,Z
01D6   0AB4          00953         incf    BUTTONPRESSED2,F                ;  是：计数上升沿
                     00954
01D7                 00955 ButtomTestEnd
01D7   0849          00956         movf    COPYPORTB,W                     ; 保存实际值以检测
01D8   00CA          00957         movwf   EDGEDETECT                      ; 下一个上升沿
                     00958
01D9   300A          00959         movlw   .10                             ; 重载去抖次数值
01DA   00C8          00960         movwf   DEBOUNCECOUNTER
```

```
01DB   29B2           00961         goto    MainLoop                        ; 并再次执行循环
                      00962
                      00963       end
存储器使用情况映射 （X = 已用， - = 未用）

0000 : X---XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX

MPASM 02.30.09 中间版本  LINSLAVE.ASM   1-27-2000  9:57:59         第 25 页


存储器使用映射 （X = 已用， - = 未用）


00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXX---- ---------------- ----------------
2000 : -------X-------- ---------------- ---------------- ----------------

所有其他存储块均未使用。

使用的程序存储字：    473
未用的程序存储字：   1575
```

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 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-16949:2002 认证。公司在 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
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

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

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
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: 852-2401-1200
Fax: 852-2401-3431

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

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

中国 – 香港特别行政区
Tel: 852-2401-1200
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: 86-755-8203-2660
Fax: 86-755-8203-1760

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

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

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

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

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: 91-80-4182-8400
Fax: 91-80-4182-8422

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

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 - 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 - 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: 34-91-708-08-90
Fax: 34-91-708-08-91

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

01/02/08