Application Note

MSPM0 器件本地互连网络 (LIN) 自动波特率的检测机制

Kalyan Gajjala, Gaurang Gupta, Ashwini Gopinath *EP-MSP*

摘要

本应用手册详细概述了 MSPM0 UART 及 UNICOMM UART 接收器 LIN 节点提供的自动波特率检测机制。所提供的信息和通过同步方法实现所需时钟频率容差的器件特别相关。在 MSPM0 UART 和 UNICOMM UART 模块中，自动波特率检测使用硬件和软件解决方案的组合来实现。

本文档首先概述了 LIN（本地互连网络）协议所规定的容差要求。然后，还介绍了如何设计 MSPM0 UART/UNICOMM UART 模块以满足这些容差限制，同时可靠地检测可变波特率。本应用手册细分了同步过程，即使在可能不准确时钟的系统中也能实现准确的数据接收。文中介绍了集成详细信息、实际实现方法和应用特定的注意事项，以帮助嵌入式系统设计人员在其项目中有效利用自动波特率检测功能。

内容

商标

1 LIN 协议简介

LIN（局域互联网络）是一种专为汽车应用设计的串行通信协议。它采用低成本标准 UART 接口，支持通过软件解决方案进行部署。LIN 协议支持的速度范围是 1 至 20Kb/S。

标准 LIN 帧分成两部分：标题和响应如图 1-1 中所示。标头由中断域、同步域和受保护标识符 (PID) 域构成。响应由数据及校验和组成。

字节间空间是指一个字节停止位末尾与下一个字节的开始位之间的时间。响应空间是 PID 域与第一个数据字节之间的时间。

每个字节域（中断域除外）的发送方式都是首先发送最低有效位 (LSB)，最后发送最高有效位 (MSB)。

本应用手册其余部分将使用的一些命名规则如下（为方便使用，根据 LIN 规范按原样使用）：

- 标称比特率： F_{Nom}
- 同步之前，响应器节点与标称位节点的偏差：F_{TOL_UNSYNC} (<+/-14%)。
- 同步后响应器节点的偏差：F_{TOL_SYNC} (<+/- 1.5%)。

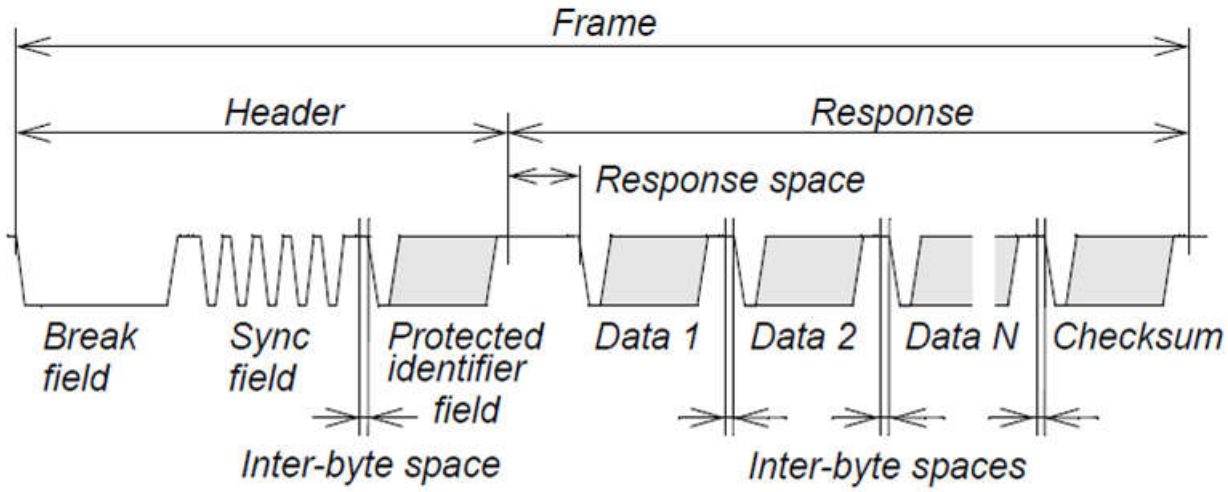


图 1-1. LIN 帧

1.1 中断域

中断域表示 LIN 帧开始，始终由命令器节点发送，在显性状态下至少具有 13 个标称位时间（零），然后是中断定界符，该定界符必须至少为一个标称位时间，如下图所示。当响应器节点看到至少 9.5 个连续显性位并且无需验证中断定界符实际上是 1 位时间长时，它可以检测到中断域。

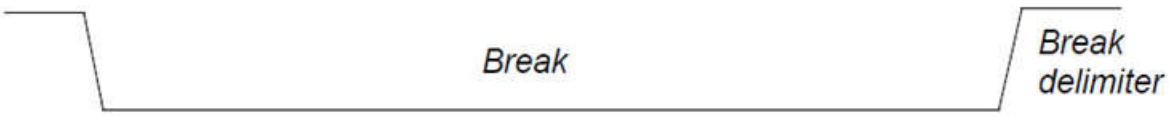


图 1-2. 中断域

1.2 SYNC 字节域

同步域是一个具有固定十六进制值 **0x55** 的字节，它使 LIN 总线上的设备能够将它们的时钟与主节点同步。值 0x55 表示一种由 1 和 0 交替的二进制模式，从而产生可预测的信号转换（边沿）。

具体工作原理如下：

- LIN 命令器会传输这种交替模式。

- 响应器节点会测量这些信号转换之间的时间，以确定主器件的精确位时序。
- 然后，响应器节点使用此测量来调整其波特率，确保它们与响应器的波特率完全同步。

图 1-3. 同步域

1.3 PID 域

LIN 中的 PID（受保护标识符）是每个 LIN 帧标头内的一个 8 位域，用于唯一标识消息，如下图所示。

它包含一个 6 位标识符 (ID)，允许 64 个唯一消息类型，其余 2 位是奇偶校验位（称为 P0 和 P1），根据 ID 位计算得出，有助于检测传输中的错误。

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT(ID1 XOR ID3 XOR ID4 XOR ID5)

PID 始终由命令器节点发送，所有响应器节点都使用它来决定是应响应消息、忽略消息还是仅侦听。

奇偶校验位允许节点检查 ID 是否正确接收，从而有助于确保数据的可靠性。

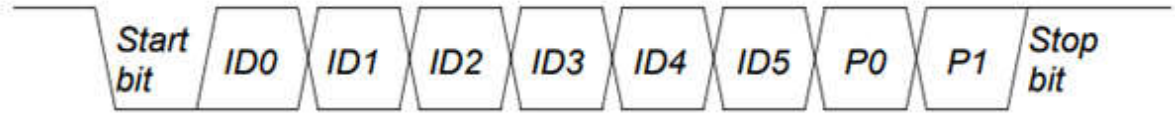


图 1-4. PID 域

1.4 数据

数据域传输命令器节点和响应器节点之间交换的实际信息，例如传感器读数或控制命令。

数据域通常包含 1 到 8 个字节，表示实际有效载荷。帧中包含的数据字节数，由帧的 PID 定义并在命令器和响应器节点之间商定。

当命令器节点通过发送报头来请求数据时，响应器节点会通过在数据域中填充相关数据并发回数据来进行回复。

数据始终后跟一个校验和字节，这有助于所有节点确认数据已正确接收且在传输过程中不会损坏。

图 1-5. 数据

1.5 校验和

校验和始终是 LIN 帧响应部分的最后一个字节，并由发送数据的节点发送。校验和计算将所有数据字节或所有数据字节与受保护标识符相加（使用模数 256 加法，因此如果总和高于 255，则“绕回”），然后反转总和，以便将校验和与所有字节相加而等于 0xFF。

2 初始波特率设置

有关 UART 功能方框图，请参阅 MSPM0 器件特定用户指南。可以使用 CLKDIV 寄存器位域进一步对 MSPM0 UART/UNICOMM UART 的输入时钟进行分频；该分频时钟被称为功能时钟或者 UART 时钟。有关 IP 的时钟速度，请参阅相应的器件数据表。

波特率除数是一个 22 位数字，由 16 位整数 (IBRD) 和 6 位小数部分 (FBRD) 组成。波特率发生器使用这两个值形成的数字来确定位采样周期。分数波特率分频器可让 UART 非常准确地生成所有标准波特率。

16 位整数加载到 UART 整数波特率分频器 IBRD 寄存器中，6 位小数部分加载到 UART 分数波特率分频器 FBRD 寄存器中。

波特率分频 (BRD) 可以使用以下公式计算：

BRD = 功能块/ (过采样 x 波特率)

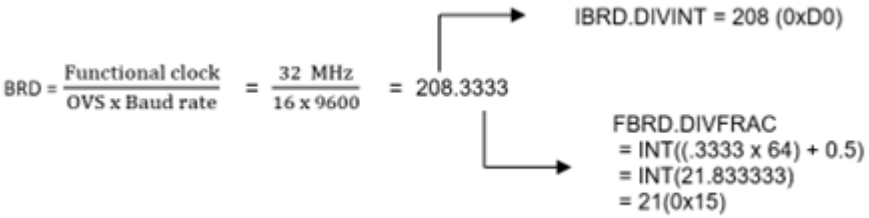
功能时钟是 UART 时钟控制逻辑的时钟输出，由 CLKSEL 和 CLKDIV 配置。过采样通过 CTL0 寄存器中的高速过采样使能 (HSE) 位进行选择，选定的过采样可以是 16、8 或 3。

- IBRD = INT (BRD)：含有波特率除数的整数部分
- FBRD = INT ((BRD – INT(BRD)*64+0.5)：含有波特率除数的残留小数部分

BRD 的整数部分被加载到 IBRD 寄存器中。必须将 6 位小数加载到 FBRD 寄存器中。

下面的示例显示了一种简单的方法来计算 9600 位/秒波特率下的 IBRD.DIVINT 和 FBRD.DIVFRAC：

- 功能时钟 = 32MHz
- 过采样 = 16
- 波特率 = 9600 位/秒



3 LIN 协议 MSPM0 UART/UNICOMM UART 的实现

MSPM0/UNICOMM 中的 UART 模块包含专用硬件功能，以支持本地互连网络 (LIN) 协议实现。这些硬件增强功能专门用于减少软件开销，并确保 LIN 通信所需的精确时序控制。为了支持本地互连网络 (LIN) 协议，在 UART 模块中实现了以下硬件增强功能：

- 由 UART 时钟计时的 16 位向上计数器 (LINCNT) (有关 UART 时钟如何生成的更多详细信息，请参阅 TRM) 。
- 计数器溢出时的中断功能 (CPU_INT.IMASK.LINOVF)。
- 具有两种可配置模式的 16 位捕获寄存器 (LINC0) ：
  - 在 RXD 下降沿捕捉 LINCNT 值。捕捉时的中断能力
  - 比较 LINCNT，匹配时可中断
- 可以配置 16 位捕捉寄存器 (LINC1) :
- 在 RXD 上升沿捕捉 LINCNT 值。捕捉时的中断能力。
- Rx 上升沿 (RXPE) 和 Rx 下降沿 (RXNE) 中断能力。

MSP 器件中的 UART 模块可以用作 LIN 命令器和 LIN 响应器。以下各部分详细介绍了为支持 LIN 协议所添加的增强功能。

备注

如果 LINC0_CAP 和 LINC0_MATCH 同时设置为 1，则 LINC0 寄存器将在匹配模式下运行，因为匹配行为将覆盖 CAP 行为。

在基于 UNICOMM 的器件中，如果 CLKDIV 需要配置为非零值，则应在配置 CLKDIV 之前配置 LINC0 MMR。 (有关更多详细信息，请参阅器件勘误表) 。

3.1 LIN 发送

命令器模式：MSPM0 UART/UNICOMM UART 生成中断、SYNC 和 PID，然后在对 PID 进行解码后发送或接收数据。

3.1.1 中断域

中断域是 LIN 通信的关键部分，用于标记 LIN 帧的开始。要在 MSP 器件中生成中断域，必须使用 LCRH.BRK 位。通过强制 UART 输出连续低电平信号，将 UART.LCRH 寄存器中的 BRK 位设置为 1 以产生中断域。

持续时间由软件/应用程序控制。

在将任何数据放入 TXDATA/FIFO 之前，必须设置 LCRH.BRK 位。中断域的持续时间至少为 13 个显性位时间，后跟一位计时器 (用于中断定界符) 。

发送中断域后，应清除 LCRH.BRK 位。

实施顺序：

- 将位 LCRH.BRK 置为有效
- 使用软件在所需持续时间内保持显性状态
- 将 LCRH.BRK 位清零以提供定界符

备注

为了确保正确生成中断域，必须在任何 TXDATA/FIFO 操作之前进行 LCRH.BRK 位配置。

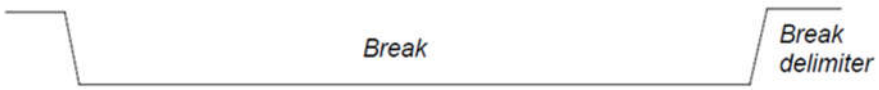


图 3-1. 中断域描述

或者，可以在比较模式下使用 LINCNT 和 LINC0 对 LIN 中断域精确计时。可以将 LINC0 配置为与中断域持续时间匹配，可以设置 LCRH.BRK 以启动中断域传输，然后可以启用 LINCNT。在 LINC0 ISR 中清除 LCRH.BRK 时，中断域结束。此外，来自其他 IP (如 GPTIMER) 的中断也可用于此目的。

3.1.2 同步域

在中断域传输且 LCRH.BRK 位取消置位后，必须将同步域 (0x55) 加载到 UART TXFIFO 中，以启动同步域传输阶段。

3.1.3 PID 域

同步域传输完成后，必须将受保护标识符 (PID) 加载到 TXDATA/FIFO 中，从而保持同步域和 PID 传输序列之间所需的字节间间隔。

3.1.4 数据域

在受保护标识符 (PID) 传输完成后，TXDATA/FIFO 中可以填充指定用于在 LIN 帧中传输的必要数据字节。

以下代码序列演示了 LIN 帧标头内的中断域、同步域和受保护标识符 (PID) 顺序传输的实现协议：

```
/* Transmit BREAK, SYNC byte, and PID */
DL_UNICOMMUART_enableLINSendBreak(uart);//initiate the process to send the break field by setting LCRH.BRK bit
delay_cycles(LIN_BREAK_LENGTH); /* Send break field conforming to the timing specifications, calculated based on the baud rate */
DL_UNICOMMUART_disableLINSendBreak(uart);//abort the break field transmission by clearing the LCRH.BRK bit
DL_UNICOMMUART_transmitDataBlocking(uart, LIN_SYNC_BYTE);//load the UART TXDATA/FIFO with SYNC field (0x55)
delay_cycles(LIN_INTER_BYTE_SPACE);//provide the inter-byte space between SYNC field and PID field
DL_UNICOMMUART_transmitDataBlocking(uart, messageTable[tableIndex].msgID);//Send the PID, in this case 0x0D
```

图 3-2. 用于传输中断域、同步域及数据字节的软件序列

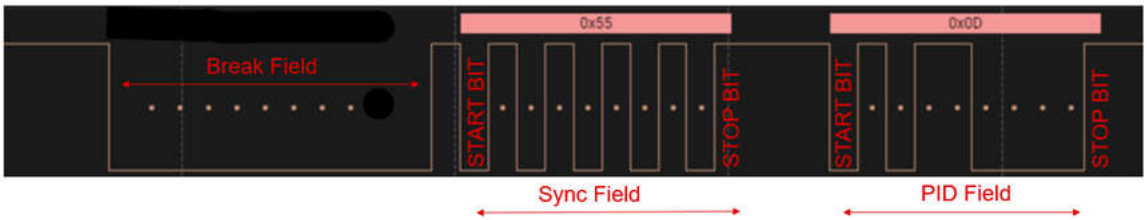


图 3-3. 由 LIN 命令器发送的中断、同步、PID 域

3.1.5 校验和

LIN 协议用取反校验和计算来实现 8 位错误检测机制。校验和计算包含连续字节求和，对于超过 0xFF 的值、进位管理、然后对最终总和进行一补码运算。

指定了两种校验和方法：

1. 经典校验和：计算仅包含数据字节
2. 增强校验和：计算包含同数据字节连接的受保护标识符 (PID)

校验和算法：

- 顺序字节累积
- 如果总和超过 0xFF，则执行加法
- 结果反转 (一补码)

实现示例：

受保护标识符：0x0D

数据域： [0xAB, 0xBC, 0xCD, 0xDE, 0xEF]

增强型校验和计算序列：

第 1 步：0x0D

初始值为 = 0x0D

第 2 步：添加 0xAB

0x0D + 0xAB = 0xB8

（由于总和小于 256，因此无需减去 255）

第 3 步：添加 0xBC

0xB8 + 0xBC = 0x174

当总和 ≥ 256 (0x100) 时，减去 255 (0xFF)

0x174 - 0xFF = 0x75

第 4 步：添加 0xCD

0x75 + 0xCD = 0x142

当 sum ≥ 256 时，减去 255

0x142 - 0xFF = 0x43

第 5 步：添加 0xDE

0x43 + 0xDE = 0x121

当 sum ≥ 256 时，减去 255

0x121 - 0xFF = 0x22

第 6 步：添加 0xEF

0x22 + 0xEF = 0x111

当 sum ≥ 256 时，减去 255

0x111 - 0xFF = 0x12

最后步骤：对结果取反

校验和 = 0xFF - 0x12 = 0xED

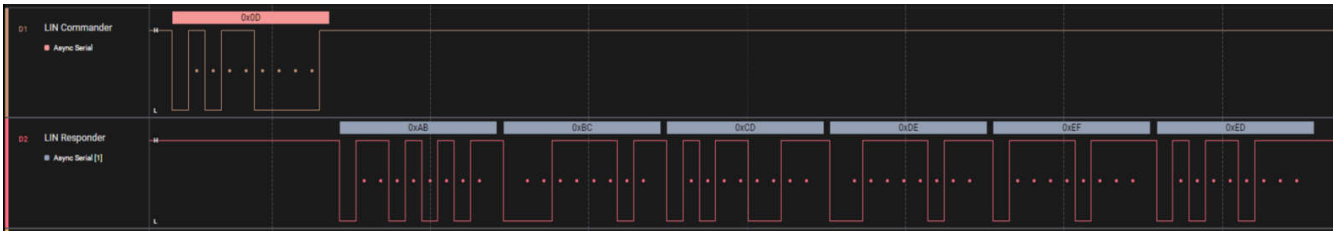


图 3-4. 校验和传输

下面随附的代码部分演示了计算及发送 LIN 校验和的过程。

```c
void sendLINResponderTXMessage( UNICOMM_Inst_Regs *uart, uint8_t tableIndex,
    uint8_t *msgBuffer, LIN_table_record_t *responderMessageTable)
{
    uint8_t locIndex;
    uint8_t checksum;
    LIN_word_t tempChksum;

    /* Disable LIN RX */
    DL_UART_Extend_disableInterrupt(uart, DL_UART_EXTEND_INTERRUPT_RX);

    tempChksum.word = responderMessageTable[tableIndex].msgID;
    tempChksum.word = tempChksum.byte[0] + tempChksum.byte[1];

    for (locIndex = 0; locIndex < responderMessageTable[tableIndex].msgSize;
         locIndex++) {
        DL_UART_Extend_transmitDataBlocking(uart, msgBuffer[locIndex]);
        tempChksum.word += msgBuffer[locIndex];
    }
    /* Calculate and send checksum */
    checksum = tempChksum.byte[0];
    checksum += tempChksum.byte[1];
    checksum = 0xFF - checksum;

    DL_UART_Extend_transmitDataBlocking(uart, checksum);
    while (DL_UART_Extend_isBusy(uart)) {
        ;
    }

    DL_UART_Extend_receiveDataBlocking(uart);

    /* Enable LIN RX */
    DL_UART_Extend_clearInterruptStatus(uart, DL_UART_EXTEND_INTERRUPT_RX);
    DL_UART_Extend_enableInterrupt(uart, DL_UART_EXTEND_INTERRUPT_RX);
}
```

图 3-5. 传输校验和的软件序列

3.2 LIN 接收

响应器模式：MSPM0 UART/UNICOMM UART 等待中断检测，然后在对 PID 进行解码后发送或接收数据。

LIN 命令器在每帧开始时发出一个中断域和一个同步域。已经添加硬件，这样 LIN 响应器软件驱动程序才能合理地检测 BREAK-SYNC 并测量必要的时序参数，以调整波特率或确定错误。

3.2.1 中断域检测

LIN 帧的接收需要利用计数器及比较模式功能实现精确的中断域检测。

配置序列：

1. 计数器初始化
   - 复位 LIN 计数器 (UARTx.LINCNT = 0)
2. 比较模式配置
   - 将计数器比较匹配模式置为有效 (UARTx.LINCTL.LINC0_MATCH = 1)
   - 使用 9.5 x Tbit 阈值配置 UARTx.LINC0
   - 启用 LINC0 匹配中断 (CPU_INT.IMASK.LINC0 = 1)
3. 计数器控制参数 (UARTx.LINCTL)
   - 启用 RXD 低电平状态计数 (LINCTL.CNTRXLOW = 1)
   - 配置下降沿计数器复位 (LINCTL.ZERONE = 1)
   - 激活计数器操作 (LINCTL.CTRENA = 1)
4. 检测功能
   - RX 上升沿中断功能 (CPU_INT.IMASK.RXPE = 1)
   - 当触发 RXPE 中断时，软件可以直接读取 LINCNT 以查看中断域时序。

可选：用户可以启用 LIN 计数器溢出中断 (CPU_INT.IMASK.LINOVF = 1)，以检测中断域过长并溢出 16 位计数器。超时可通过以下公式计算得出：$t_{Timeout}$**= 216/ UART 时钟**

3.2.2 同步域验证

同步域验证对于确保 LIN 帧标头的精确计时精度和确定命令器的波特率参数至关重要。验证序列成功确认了多个标准：正确的中断域检测、精确的通信时序、准确的保护标识符 (PID) 接收及整体帧同步完整性。

同步域包含一个预定义 0x55 字节模式 (01010101)，专门用于帮助实现以下目的：

- 通过交替位模式实现精确的时序参考
- 四种不同的位时间测量机会
- 用于验证的确定性转换间隔

此结构化模式让接收节点能够：

- 确定实际通信参数
- 根据需要实施波特率调整
- 实现和主时序基准的同步

验证中断域后，系统使用 LINC1 捕获寄存器启动同步域测量，该寄存器在 RX 上升沿捕获 LINCNT 值。LINCNT 计数器配置成在下降沿复位并在 RX LOW 状态期间递增。LINC1 捕获操作和 RX 上升沿中断在每个上升沿转换之时触发。在中断服务例程期间，软件通过 LINC1 寄存器值分析各个位时序参数，验证时序规格并在需要时实现波特率调整。

图 3-6. 同步域 - 0x55

3.3 LIN 收发器

本应用手册使用 TLIN2029A-Q1 评估模块 (EVM) 作为外部 LIN 收发器。下图显示了 MSPM0 命令器和响应器如何与 TLIN2029A-Q1 收发器连接的方框图。

有关收发器和 LIN 命令器/响应器之间原理图连接的更多详细信息，请参阅 TLIN2029-Q1 EVM 用户指南。

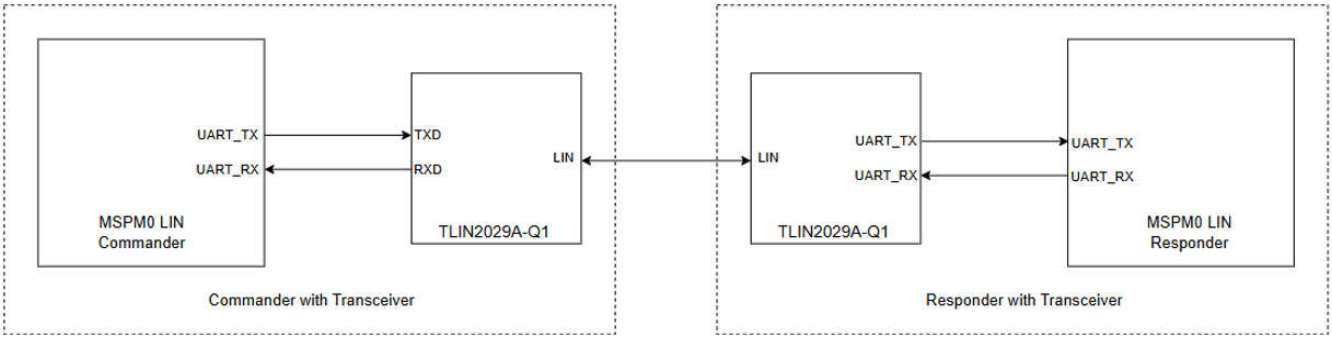


图 3-7. MSPM0 命令器和响应器和 TLIN2029A-Q1 收发器连接的方框图

4 自动波特率检测

LIN 中的自动波特率检测过程是每次 LIN 帧启动时，响应器节点都会自动识别并调整为命令器的波特率。

命令节点在每帧开始时发送一个特殊的同步域，该域始终为字节 0x55（二进制文件 01010101）。

响应器节点可以通过使用 MSPM0/UNICOMM 寄存器测量同步字节中的位时间 (Tbit) 来计算当前波特率（请参阅节 3.2.2）。

4.1 使用 MSPM0 UART/UNICOMM UART 测量位宽的过程

响应器节点可计算当前波特率，方法是测量同步字节中每个上升沿的位时间 (Tbit)，上升沿的距离为 1、3、5、7 和 STOP 位时间，如下图所示。

在自动波特率检测中，要计算同步域位时序，可在 MSPM0/UNICOMM 上使用以下寄存器：

1. 在检测到有效的中断域后，将 LIN 计数器初始化为 0 (LINCNT = 0)。
2. 在 RX 上升沿启用中断 (CPU_INT.IMASK.RXPE = 1)
3. 设置 LIN 计数控制 (LINCTL) 寄存器
   a. 在 RX 下降沿启用 LIN 计数器清零 (LINCTL.ZERONE = 1)
   b. 当 RX 上的信号为低电平时启用计数 (LINCTL.CNTRXLOW = 1)
   c. 在 RX 上升沿启用 LIN 计数器捕捉 (LINCTL.LINC1CAP = 1)
   d. 启用 LIN 计数器 (LINCTL.CTRENA = 1)

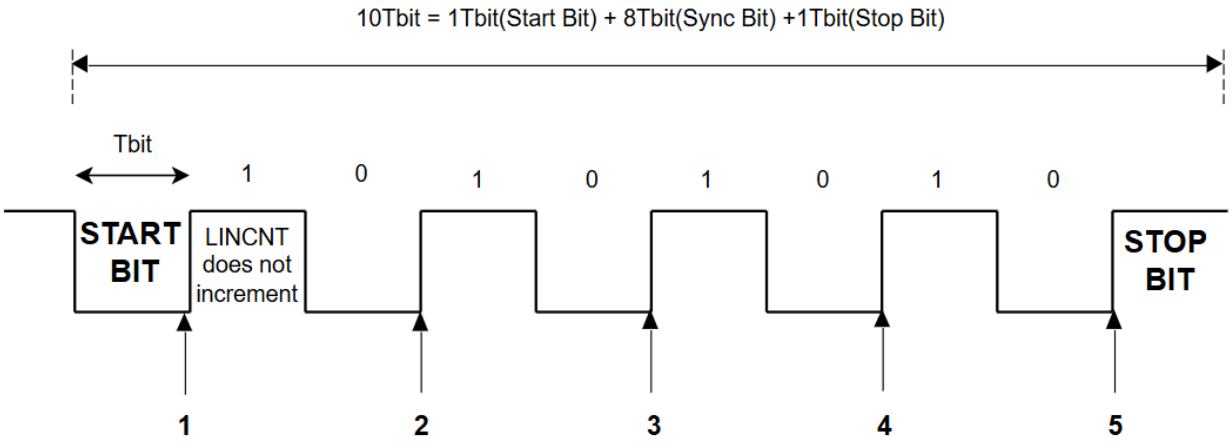


图 4-1. LIN 同步域验证

在同步域期间，在 RX 线的每个上升沿执行的操作如下。

1. LIN 计数器在每个 RX 下降沿复位为 0，并在 RX 为低电平时在 Tbit 持续时间内开始计数。
2. LINCNT 值将由 LINC1 寄存器在每个 RX 上升沿捕获。
3. 在五次迭代中触发 RX 上升沿中断 (RXPE)。
   - 在每次 RX 上升沿中断服务例程 (ISR) 迭代中，都会读取 LINC1 捕获寄存器。捕获的值表示 Tbit 时间，因为 LIN 计数器配置为仅在 RX 为低电平时计数。

4.2 计算正确的波特率

如果响应器的时钟在同步前以 32MHz（标称频率 F_{Nom}）运行，且命令器节点以 9600 波特率发送同步域。软件可以通过以下步骤计算正确波特率分频值：

对五个 LINC1 捕获的值求平均值。

$$\text{Average Bit Time (Tbit)} = \frac{\text{Total Bit time}}{5} = \frac{16665}{5} = 3333 \text{ functional clock cycles} \quad (1)$$

计算出的 Tbit 指示每个位时间内的功能时钟周期数。用户必须使用计算值更新 IBRD/FBRD 寄存器，以便在同步后保持波特率容差 (F_{TOL_SYNC})。

以下流程图示例显示了一种简单的方法,可在计算 Tbit 持续时间后确定 IBRD（整数波特率除数）和 FBRD（分数波特率除数）的值。

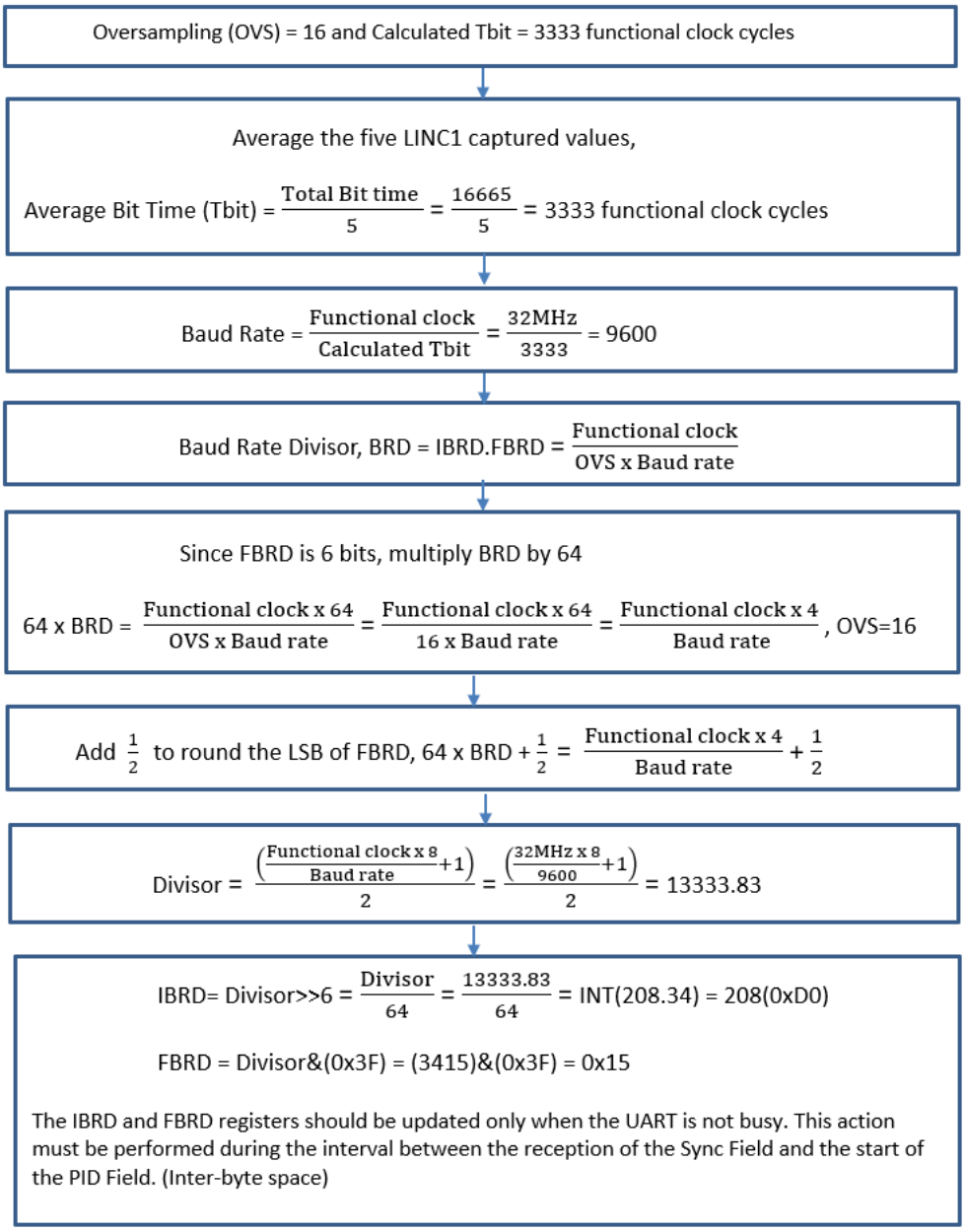


图 4-2. 标称频率下 LIN 波特率分频值计算的流程图 (32MHz)

4.2.1 响应器节点处的晶体误差

如果同步前响应器的时钟运行速度比标称速率慢 14%，特别是在 27.52MHz 下（而不是以 9600 的波特率运行的预期 32MHz 下），计算出的 Tbit 将为 2867 个时钟周期（存在错误时钟），而不是对实际时钟预期的 3333 个周期。

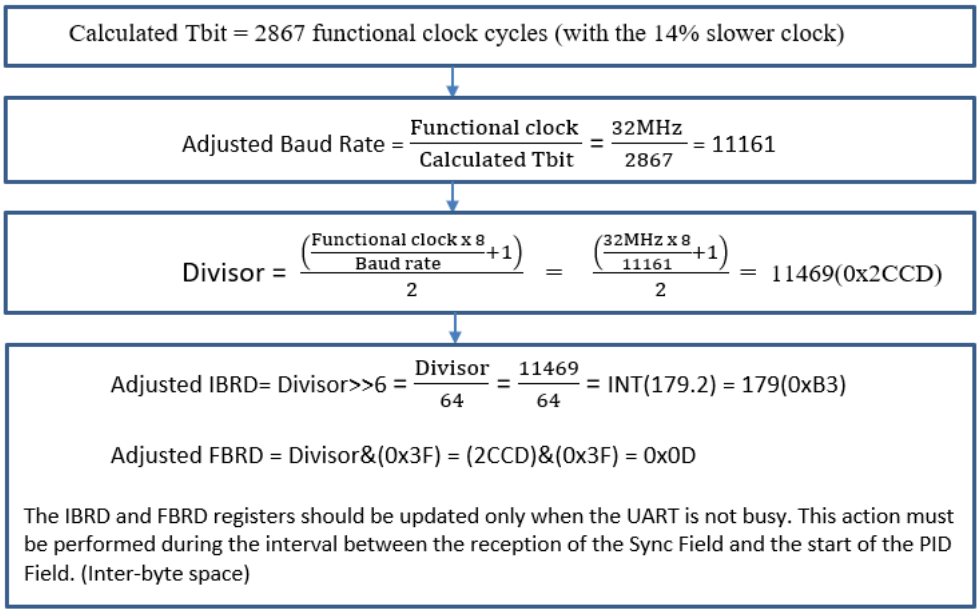


图 4-3. LIN 波特率分频值计算流程图，其中时钟速度降低 14% (27.52MHz)

如果同步前响应器的时钟运行速度比标称速率快 14%，特别是在 36.48MHz 下（而不是以 9600 的波特率运行的预期 32MHz 下），计算出的 Tbit 将为 3800 个时钟周期（存在错误时钟），而不是对实际时钟预期的 3333 个周期。

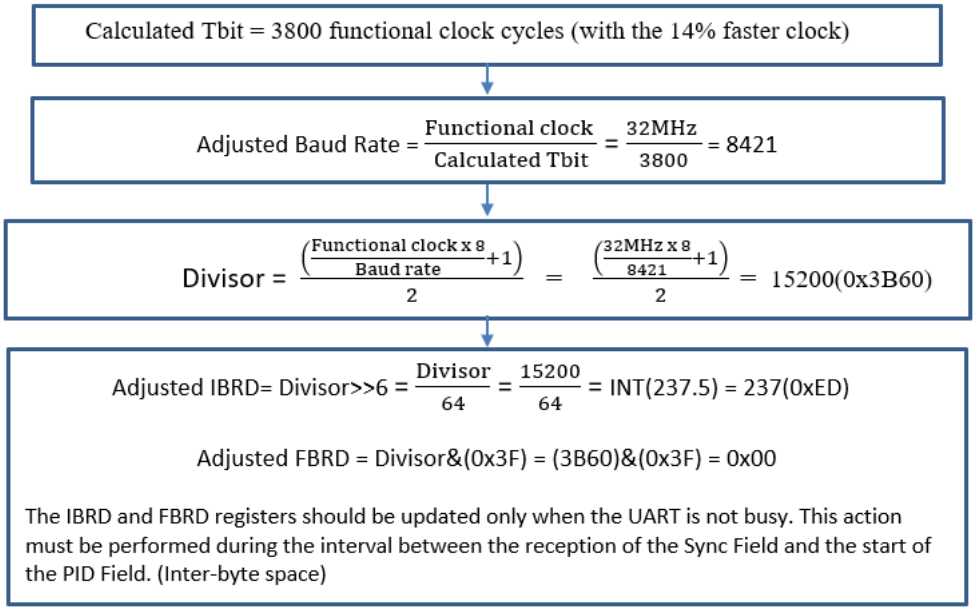


图 4-4. LIN 波特率分频值计算流程图，其中时钟速度提高 14% (36.48MHz)

以下软件代码介绍了上升沿 ISR 内的波特率检测及配置。

```
case DL_UART_EXTEND_IIDX_RXD_POS_EDGE:
          /* Signals the positive edge of a sync field segment. */
       if (gStateMachine == LIN_STATE_SYNC_FIELD_POS_EDGE)
         {
              gBitTimes[gNumCycles].posEdge =
                  DL_UART_Extend_getLINRisingEdgeCaptureValue(LIN_0_INST);
              /* Validation check of the timing of the sync field segment.
               * Finding an invalid sync bit stores each bit time to
               * calculate new baud rate */
              if (gBitTimes[gNumCycles].posEdge > ((gLin0TbitWidthVar * 95) / 100) &&
                  gBitTimes[gNumCycles].posEdge < ((gLin0TbitWidthVar * 105) / 100))
                {
                 gNumCycles++;
                }
               else if (!gFirstSyncBit)
                 {
                  gTotalBitTime = gTotalBitTime + gBitTimes[gNumCycles].posEdge;
                  gNumSyncErrors++;
                 }
               else
                {
```

```
          gFirstSyncBit = false;
        }
          /* Only 5 segments of a sync field. */
       if ((gNumSyncErrors + gNumCycles) ==LIN_RESPONDER_SYNC_CYCLES)
        {
          DL_UART_Extend_enableInterrupt(LIN_0_INST, DL_UART_EXTEND_INTERRUPT_RX);
          DL_UART_Extend_disableInterrupt(
              LIN_0_INST, DL_UART_EXTEND_INTERRUPT_RXD_NEG_EDGE);

          /* Track new and previous baud rate for validation when
           * increasing baud rate. Ensures that resets to deal with
           * overrun errors happen on the appropriate frame.
           * Reset all variables relevant to sync field */
          if (gNumCycles == LIN_RESPONDER_SYNC_CYCLES)
            {
              gPrevBaudRate = gCurrBaudRate;
              gAutoBaudUsed = false;
            }
          gNumCycles     = 0;
          gNumSyncErrors = 0;
          gTotalBitTime  = 0;
          gFirstSyncBit  = true;

          /* If 4 sync errors are detected, update baud rate given
          * autobaud is enabled*/
        }
       else if ((gNumSyncErrors == AUTO_BAUD_THRESHOLD) &&AUTO_BAUD_ENABLED)
        {
          averageBitTime   = gTotalBitTime / gNumSyncErrors;
          measuredBaudRate = LIN_0_INST_FREQUENCY / averageBitTime;

          // Wait for UART Busy bit to go LOW
          while(DL_UART_isBusy(LIN_0_INST));

          gLinResponseLapseVar = LIN_0_INST_FREQUENCY / (2 * measuredBaudRate);
          gLin0TbitWidthVar = averageBitTime;
          // Configure new calculated baud rate
         DL_UART_configBaudRate(LIN_0_INST,LIN_0_INST_FREQUENCY,measuredBaudRate);
         DL_UART_Extend_setLINCounterCompareValue(LIN_0_INST,
              gLin0TbitWidthVar * LIN_0_TBIT_COUNTER_COEFFICIENT);


          gPrevBaudRate = gCurrBaudRate;
          gCurrBaudRate = measuredBaudRate;
          gAutoBaudUsed = true;
```

```
            gStateMachine = LIN_STATE_SYNC_FIELD_NEG_EDGE;
          }
        else
          {
            gStateMachine = LIN_STATE_SYNC_FIELD_NEG_EDGE;
          }
      }
      break;
```

5 同步后波特率偏差

如果同步前响应器的时钟运行速度比标称速率慢 14%，则功能时钟变为 27.52MHz 而非预期的 32MHz。同步后，IBRD 和 FBRD 寄存器值分别为 0xB3 和 0x0D。

调整后的波特率为 = $\frac{\text{Functional clock}}{\text{OVS x IBRD.FBRD}} = \frac{27.52\text{MHz}}{16 \times 179.13}$ = 9601.1

预期波特率为 9600，相应的预期位时间 (Tbit) 为 104.16µs。

同步后实际计算出的波特率为 9601.1，因此实际 Tbit 为 104.15µs。

$$\text{Percentage of error in Baud Rate post synchronization} = \frac{(\text{Expected Tbit} - \text{Actual Tbit})}{\text{Actual Tbit}} \times 100$$

$$= \frac{(104.16 - 104.15)}{104.16} \times 100 = 0.01\%$$

同步后，响应器节点的波特率偏差为 0.01%，这完全在 LIN 规范的允许容差限制 (FTOL_SYNC < ±1.5%) 范围内。

6 参考资料

1. MSPM0 G 系列 80MHz 微控制器技术参考手册（修订版 C）
2. LIN-2.2 规格文档
3. TLIN2029-Q1 EVM 用户指南
4. 附有 CAN-FD 接口的 MSPM0Gx51x 混合信号微控制器数据表

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月