Application Note

LIN 的基础知识和在 MSPM0 上的实现

Zoey Wei

摘要

本应用说明重点介绍使用 MSPM0 实现 LIN 基本功能的方法。具体而言，将介绍 MSPM0 如何与 LIN 驱动程序配合实现协议层和物理层功能，以帮助快速开发软件项目。此外，请注意 MSPM0 软件层仅支持简单的 LIN 通信。

内容

商标

1 简介

LIN（本地互连网络）总线是一种基于 UART/SCI（通用异步收发器/串行通信接口）的低成本串行通信协议。由于成本低廉，它作为 CAN 的子线被广泛应用于汽车领域。如 图 1-1 所示，LIN 通信采用单命令器和多响应器架构，MCU 使用 UART 接口结合 LIN 收发器在节点之间进行通信。

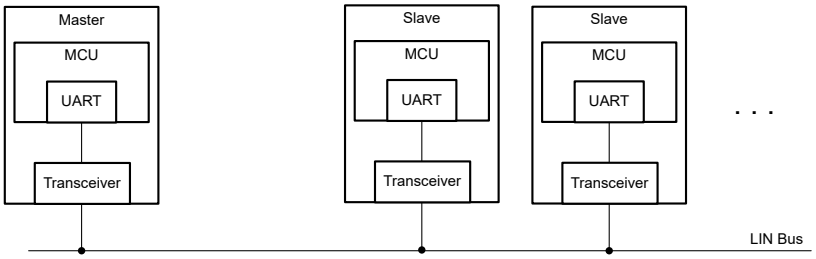

图 1-1. LIN 网络

与大多数网络协议类似，LIN 被正式定义为一个多层系统，从物理接口到应用层各不相同，如 图 1-2 所示。节点应用层向下一层传输信号和消息，并通过协议层将它们封装成帧格式，然后通过 LIN 总线传输给其他节点。

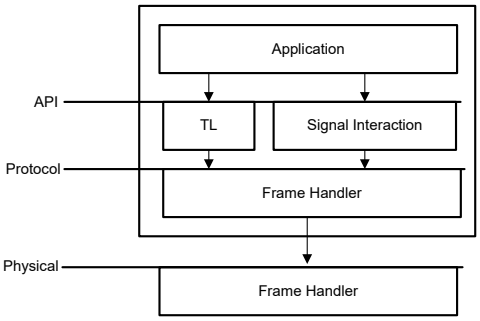

图 1-2. LIN 节点的组成

本应用说明重点介绍使用 MSPM0 实现 LIN 基本功能的方法。具体而言，介绍了 MSPM0 如何与 LIN 驱动程序配合实现协议层和物理层功能，以帮助快速开发软件项目。

2 MSPM0 如何支持 LIN 功能

2.1 时钟

LIN 规范规定了命令器-响应器节点的时钟精度，具体反映在比特率规范中，如 表 2-1 所示。

表 2-1. LIN 比特率要求

比特率容差	名称	$\Delta F / F_{Nom}$ [1]
命令器节点（与标称比特率的偏差）	$F_{TOL\_RES\_MASTER}$	<±0.5%
未使用同步的响应器节点（与标称比特率的偏差）	$F_{TOL\_RES\_SLAVE}$	<±1.5%
同步前，响应器节点比特率与标称比特率的偏差；适用于使用同步和直接中断检测的节点。	F_{tol_unsync}	±14%
同步后，响应器节点比特率相对于命令器节点比特率的偏差	F_{TOL_SYNC}	<±2%
对于任何两个节点之间的通信（例如，从一个响应器到另一个响应器的数据流），比特率差异不得超过 $F_{TOL\_SL\_to\_SL}$	$F_{TOL\_SL\_to\_SL}$	<±2%

(1) LIN 总线上使用的特定比特率定义为标称比特率 F_{Nom}

表 2-2 显示了 MSPM0 的时钟规格。

表 2-2. MSPM0 的时钟规格 (T =25℃)

串（联）	时钟	支持外部晶体振荡器
MSPM0G	精度高达 0.7% 的内部 4MHz 至 32MHz 振荡器 (SYSOSC)[1]	外部 4MHz 至 48MHz 晶体振荡器 (HFXT)
	精度为 ±3% 的内部 32kHz 低频振荡器 (LFOSC)	外部 32kHz 晶体振荡器 (LFXT)
MSPM0L	精度为 ±0.7% 的内部 4MHz 至 32MHz 振荡器 (SYSOSC)	不支持
	精度为 ±3% 的内部 32kHz 低频振荡器 (LFOSC)	
MSPM0C	精度高达 ±1% 的内部 24MHz 振荡器 (SYSOSC)	支持（仅支持 20 引脚）
	精度为 ±3% 的内部 32kHz 低频振荡器 (LFOSC)	

(1) MSPM0G 系列支持高达 80MHz 的 PLL

2.2 LIN 硬件

为了支持本地互连网络 (LIN) 协议，在 UART0 模块中实现了以下硬件增强功能：

- 由 UART 时钟计时的 16 位加法计数器 (LINCNT)。
- 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 位捕捉寄存器 (LINC0)
  - 在 RXD 下降沿捕捉 LINCNT 值。捕捉时的中断能力。
  - 比较 LINCNT 与匹配时的中断能力。
- 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。

此外，MSPM0 还具有支持 LIN 通信的大型寄存器。例如，作为命令器时，有一个 LCRH.BRK 寄存器可以使 UART.TXD 持续发送低电平。对于响应器，LINCNT 寄存器可以帮助获取中断字段的时间。有关更多详细信息，请参阅 *MSPM0 G 系列 80MHz 微控制器*技术参考手册。

2.3 SDK 中的 LIN 演示代码

为了帮助快速、方便地开发 LIN 通信，TI 在 *SDK* 中提供了响应器和命令器的演示代码。您可以从 *MSPM0-SDK 软件开发套件*下载 MSPM0 的所有代码示例。该演示代码将 UART 配置为 LIN 命令器或响应器，并使用增强型校验和演示了 LIN 2.0 数据包的基本传输和接收。一些关键特性：

- 波特率：19200bps
- 发送数据时，通过预定义符号选择中断或轮询模式
- 接收数据时执行超时功能

2.3.1 LIN 命令器

LIN 命令器演示代码的主要功能是在 LIN 协议帧中发送不同的命令。当按下两个按钮时，会紧急发送 PID 0x39 和 0x08。其中一个用于点亮响应器的 LED，另一个用于从响应器接收数据。

图 2-1 展示了 LIN 命令器演示代码的流程。

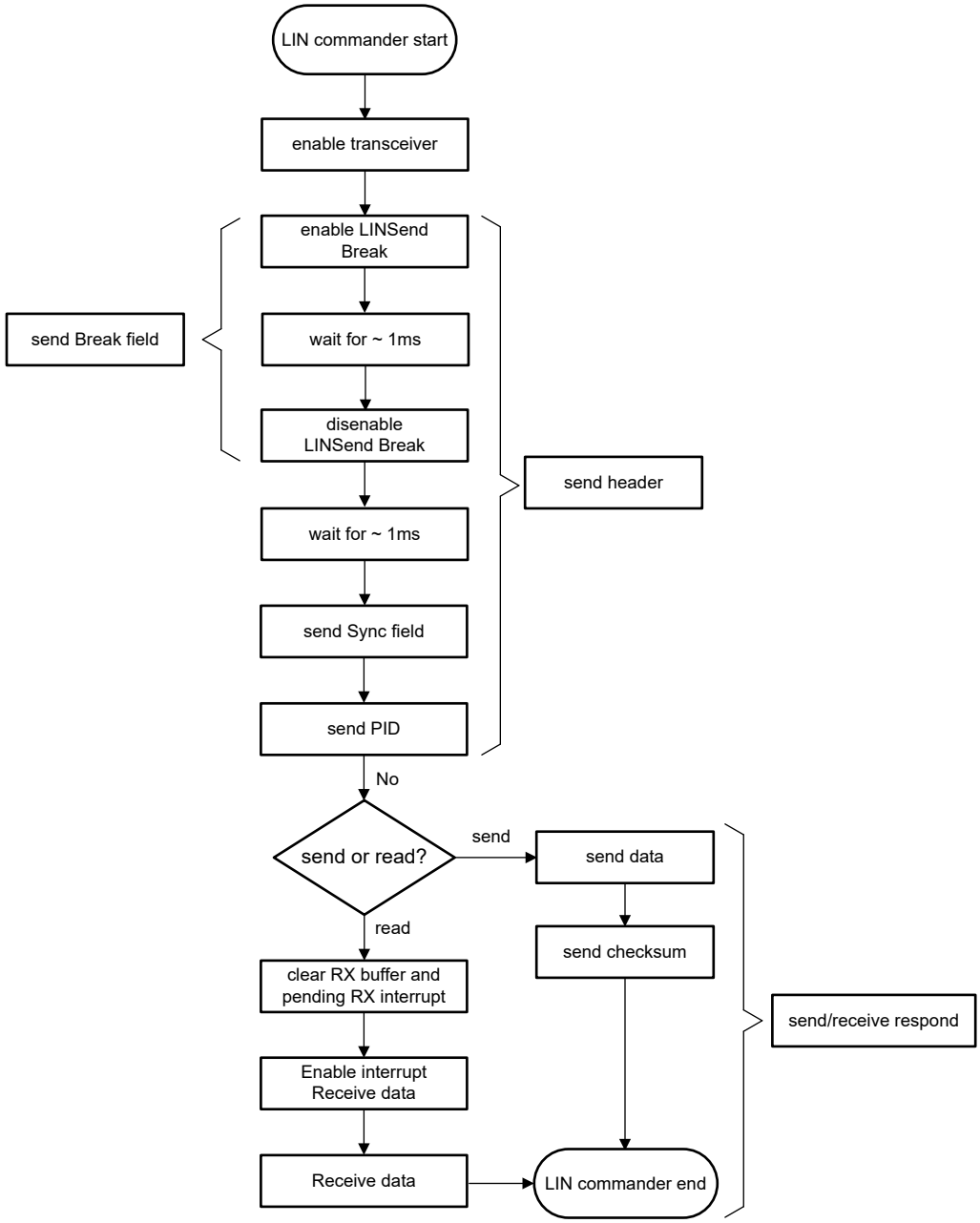

图 2-1. LIN 命令器演示代码的流程

为了初始化硬件，使用了 TI 系统配置工具 (SysConfig) 生成 UART 的配置代码，如 UART 时钟、引脚配置等。在此演示代码中，我们按照 LIN 规范要求，选择了 19200 波特率。

表 2-3 展示了 LIN 命令器项目的摘要，其中包括主要使用的定义和函数。

表 2-3. LIN 命令器项目主要内容

名称	任务	说明	位置
LIN_TABLE_INDEX_PID_xx	定义	每个帧的 PID	lin_command.c
LIN_MESSAGE_NOT_FOUND	定义	0xFF。未找到消息时的 UART LIN 值	lin_config.h
LIN_SYNC_BYTE	定义	0x55。同步字节的 UART LIN 值	lin_config.h
LIN_BREAK_LENGTH	定义	0x08。UART LIN 中断长度设为 1ms	lin_config.h
SYSCFG_DL_init()	函数	初始化由 Sysconfig 生成的外设	ti_msp_dl_config.c
DL_UART_enableLINSendBreak(UART_Regs *uart)	函数	启用发送中断。启用后，当前字符传输完毕后，TXD 信号持续输出低电平	dl_uart.c
DL_UART_disableLINSendBreak(UART_Regs *uart)	函数	禁用发送中断	dl_uart.c
LIN_Commander_transmitMessage(UART_Regs *uart, uint8_t tableIndex, uint8_t *buffer, LIN_table_record_t *messageTable)	函数	LIN 传输消息	lin_config.c
LIN_processMessage_Rx()	函数	在收到消息时进行处理	Lin_commander.c

2.3.2 LIN 响应器

LIN 命令器演示代码的主要功能是接收命令器发出的命令并执行相应的指令。在此演示代码中，响应器不进行时钟同步，而是使用自时钟，并且只检查同步字节是否正确。此外，没有超时错误检测。

图 2-2 展示了 LIN 响应器演示代码的流程。

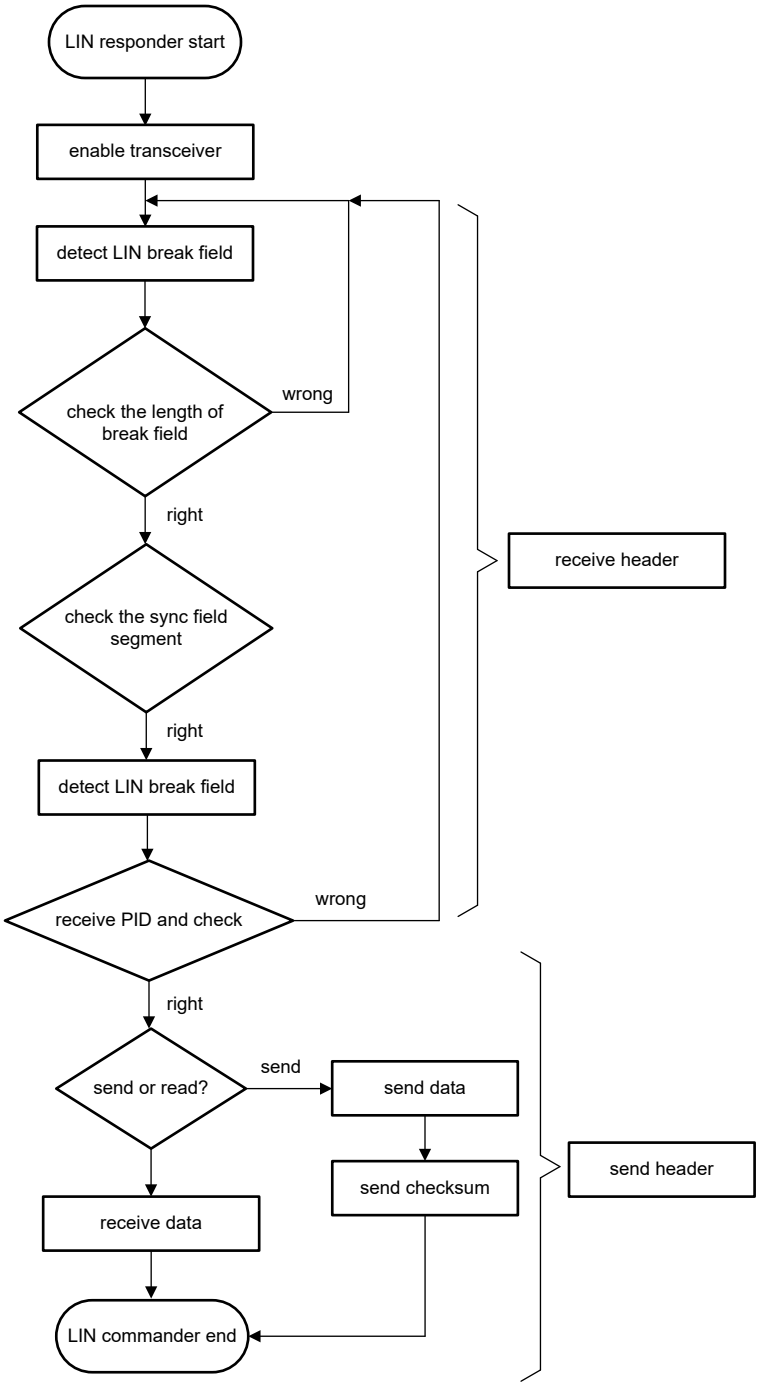

图 2-2. LIN 响应器演示代码流程

在该演示代码中，使用了一个状态机来帮助接收命令和给出响应，如 图 2-3 所示。当中断发生时，将通过状态标志决定下一个未执行的状态，并跳转到该状态。

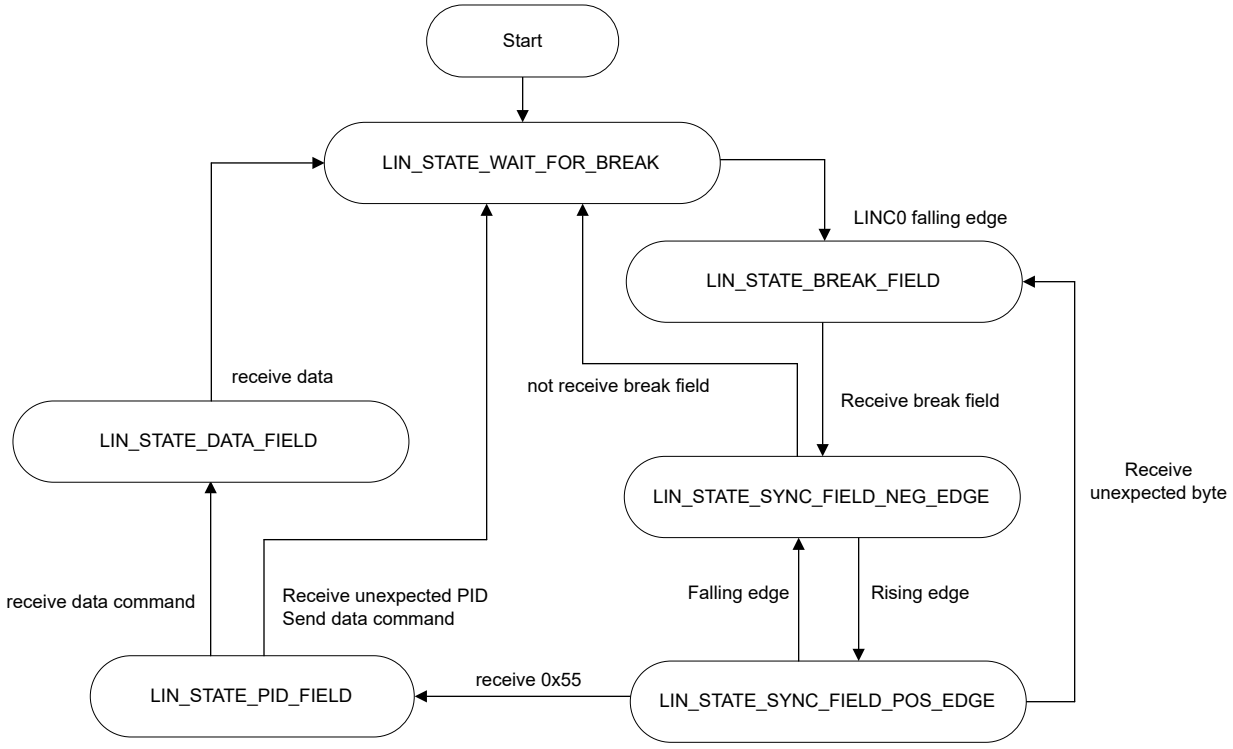

图 2-3. LIN 响应器演示代码状态机

在该演示代码中，LIN 硬件通过 Sysconfig 进行初始化。除了引脚配置外，还需要设置中断和基本寄存器。LIN 计数器用于检测和验证中断字段。此外，同步字段通过 LINC0、LINC1 的计数和中断功能实现。有关详细设置，请参阅代码示例中的 .syscfg 文件。

表 2-4 是 LIN 响应器项目的摘要，其中包括主要使用的定义和函数。

表 2-4. LIN 响应器项目主要内容

名称	任务	说明	位置
LIN_RESPONDER_SYNC_CYCLES	定义	5.同步验证中的周期数	lin_config.h
LIN_MESSAGE_NOT_FOUND	定义	0xFF。未找到消息时的 UART LIN 值	lin_config.h
LIN_SYNC_BYTE	定义	0x55。同步字节的 UART LIN 值	lin_config.h
LIN_RESPONSE_LAPSE	定义	PID 停止位和数据传输开始位之间的延迟周期数	lin_config.h
SYSCFG_DL_init()	函数	初始化由 Sysconfig 生成的外设	ti_msp_dl_config.c
DL_UART_Extend_getLINCounterValue(UART_Regs *uart)	函数	获取 LIN 计数器值	dl_uart.c
DL_UART_Extend_getLINRisingEdgeCaptureValue(UART_Regs *uart)	函数	获取 LINC1 计数器值	dl_uart.c
DL_UART_Extend_getLINFallingEdgeCaptureValue(UART_Regs *uart)	函数	获取 LINC0 计数器值	dl_uart.c
DL_UART_Extend_receiveData(UART_Regs *uart)	函数	读取 RX FIFO 中的数据	dl_uart.c
setLINResponderRXMessage(UART_Regs *uart, uint8_t data, volatile LIN_STATE *gStateMachine)	函数	LIN 执行相应操作	lin_responder.c

3 LIN 通信的实现

3.1 硬件连接

MCU 需要配合 LIN 收发器使用，因为 LIN 收发器可以将 UART 信号转换为 LIN 信号。

这里以 LP-MSPM0G3057 和 TLIN2029EVM 为例。硬件连接如 图 3-1 所示。这两个连接形成一个通信单元，可以连接到另一个单元，如带有收发器的 MCU、连接到 PC 的 LIN 分析仪。

1. 将 MSPM0G3507 的 UART 引脚连接至 TLIN2029。
2. 将 MSPM0G3507 的电源引脚（3.3V 和 GND）连接至 TLIN2029。
3. 将作为使能引脚的 MSPM0G3507 PB15 连接至 TLIN2029 的 ENABLE 引脚。
4. 使用外部 12V 电源为 TLIN 的 VBAT 引脚供电。如果其他已连接的通信单元（例如 LIN 分析仪）已经连接了 12V 电源，则可以忽略此步骤。
5. 将 TLIN2029 的 VBAT、LIN 和 GND 引脚连接至响应器或命令器硬件。
6. 为 MCU 供电。

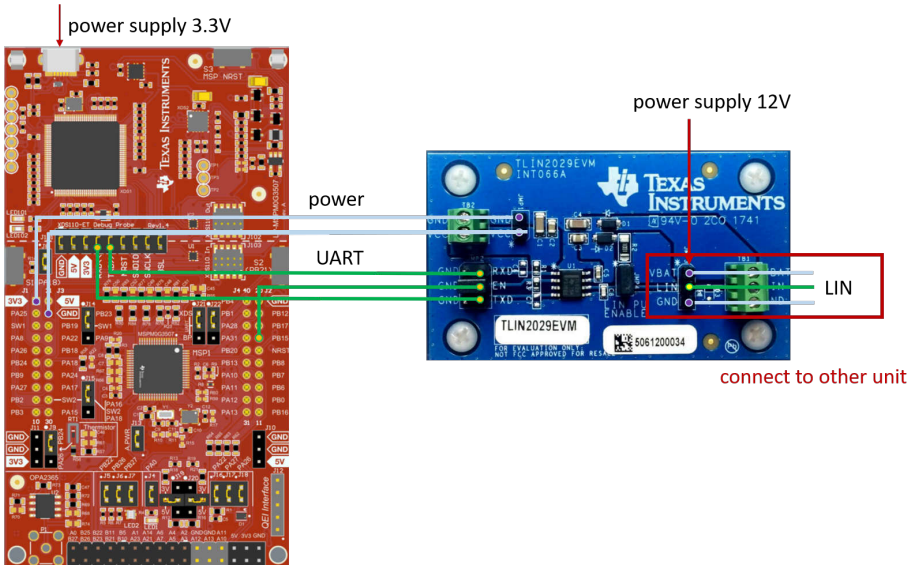

图 3-1. MSPM0 和 LIN 收发器之间的硬件连接

3.2 测试结果

接下来，将 MSPM0 分别作为命令器和响应器进行测试，可以使用 CAN&LIN 分析仪通过 LIN 与 MSPM0 通信。

3.2.1 命令器

在本例中，MCU 作为命令器，带有 CAN/LIN 分析功能的 PC 作为响应器。波特率为 19200。

当选择 Button1 使 MCU 发送 0x39(PID) 时，结果如 图 3-2 所示。如 图 3-2 所示，主机可以成功接收 MCU 发送的数据，这也可以从波形图 (图 3-3) 中确认。

ID[PID]	DATA(Hex)	Check(Hex)	Check Type	Data Type	Timestamp	Channel
39 [39]	02 02 03 04 05 06 07 09	A0	Enhanced C…	Slave Read	0.000	LIN1
39 [39]	03 02 03 04 05 06 07 0A	9E	Enhanced C…	Slave Read	0.899	LIN1
39 [39]	04 02 03 04 05 06 07 0B	9C	Enhanced C…	Slave Read	2.100	LIN1
39 [39]	05 02 03 04 05 06 07 0C	9A	Enhanced C…	Slave Read	2.532	LIN1
39 [39]	06 02 03 04 05 06 07 0D	98	Enhanced C…	Slave Read	2.933	LIN1

图 3-2. MCU 作为命令器传输数据的结果

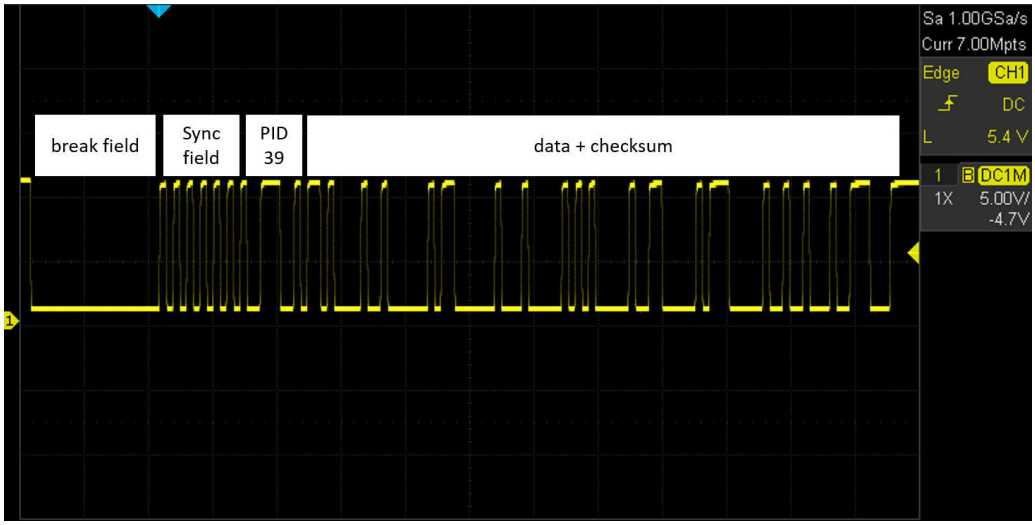

图 3-3. MCU 作为命令器传输数据的波形

当按下 button2 使 MCU 发送 0x08(PID) 时，响应器可以向 MCU 发送数据。如 图 3-4 所示，响应器传输 0x11、0x22、0x33 和 0x44，MCU 成功接收到这些数据。但在本例中，PC 中的校验模式是正常模式，因此校验和无法与 MCU 匹配，导致回调函数无法工作。但是，接收数据仍可存储在数组中。

ID[PID]	DATA(HEX)	Check(Hex)	Check Type	Data Type	Timestamp	Channel	Event
08 [08]	00 11 22 33 44	4D	Enhanced C…	Slave Write	0.000	LIN1	
08 [08]	00 11 22 33 44	4D	Enhanced C…	Slave Write	0.596	LIN1	
08 [08]	00 11 22 33 44	4D	Enhanced C…	Slave Write	1.156	LIN1	
08 [08]	00 11 22 33 44	4D	Enhanced C…	Slave Write	1.777	LIN1	

Set slave response data

Device Channel: UTA0503-[5300238F][LIN1]-Slave

Select	Data Type	Check Type	ID(Hex)	Data(Hex)	Response Mode
1	Slave Read	Standard Check	00		Loop Response
2	Slave Read	Standard Check	01		Loop Response
3	Slave Read	Standard Check	02		Loop Response
4	Slave Read	Standard Check	03		Loop Response
5	Slave Read	Standard Check	04		Loop Response
6	Slave Read	Standard Check	05		Loop Response
7	Slave Read	Standard Check	06		Loop Response
8	Slave Read	Standard Check	07		Loop Response
9	Slave Write	Enhanced Check	08	00 11 22 33 44	Loop Response
10	Slave Read	Standard Check	09		Loop Response
11	Slave Read	Standard Check	0A		Loop Response
12	Slave Read	Standard Check	0B		Loop Response
13	Slave Read	Standard Check	0C		Loop Response
14	Slave Read	Standard Check	0D		Loop Response
15	Slave Read	Standard Check	0E		Loop Response
16	Slave Read	Standard Check	0F		Loop Response
17	Slave Read	Standard Check	10		Loop Response

Select All | Invert Select | Set data

图 3-4. MCU 作为命令器接收数据的结果

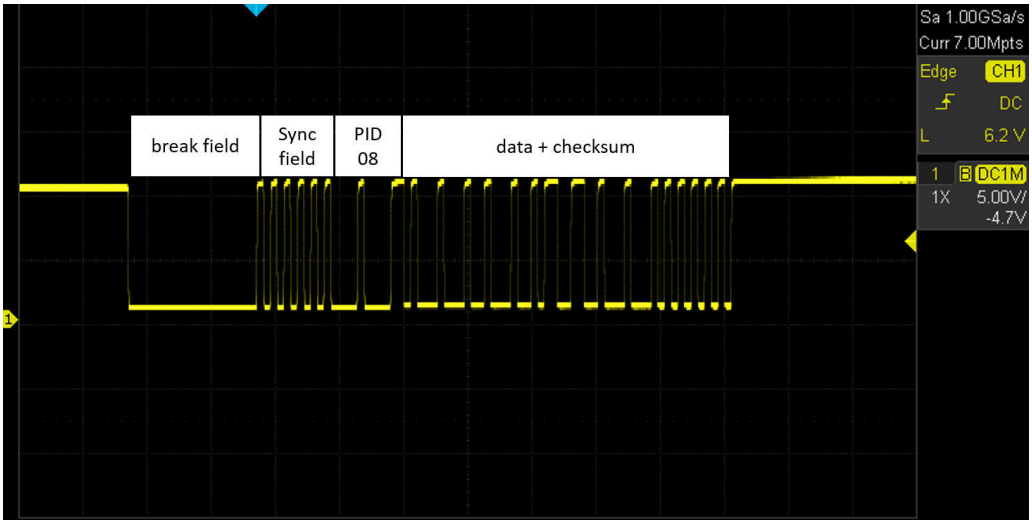

图 3-5. MCU 作为命令器接收数据的波形

gCommanderRXBuffer	unsigned char[8]	[0x11 '\x11',0x22 '"',0x33 '3',0x44 'D',0x55 'U'...]...	0x20200038
[0]	unsigned char	0x11 '\x11' (Hex)	0x20200038
[1]	unsigned char	0x22 '"' (Hex)	0x20200039
[2]	unsigned char	0x33 '3' (Hex)	0x2020003A
[3]	unsigned char	0x44 'D' (Hex)	0x2020003B
[4]	unsigned char	0x55 'U' (Hex)	0x2020003C
[5]	unsigned char	0x00 '\x00' (Hex)	0x2020003D
[6]	unsigned char	0x00 '\x00' (Hex)	0x2020003E

图 3-6. gCommanderRXBuffer 结果

3.2.2 响应器

在本例中，MSPM0 作为响应器。演示代码实现了以下功能：如果接收到 0x39/0xBA/0xFB，MCU 可以从命令器接收数据。如果接收到 0x08/0x49/0x0D PID，MCU 可以向命令器传输数据。

当命令器发送 0x3B（PID 为 0xFB）时，MCU 将从主机接收数据。如 图 3-7、图 3-8 和 图 3-9 所示，主机成功传输了数据，这些数据可以在调试模式下从 MCU RAM 中读取。

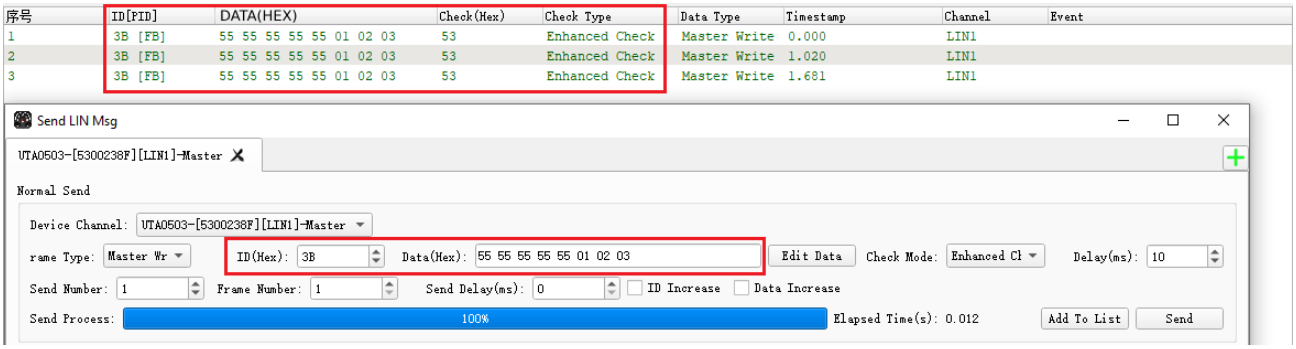

图 3-7. MCU 作为响应器接收数据的结果

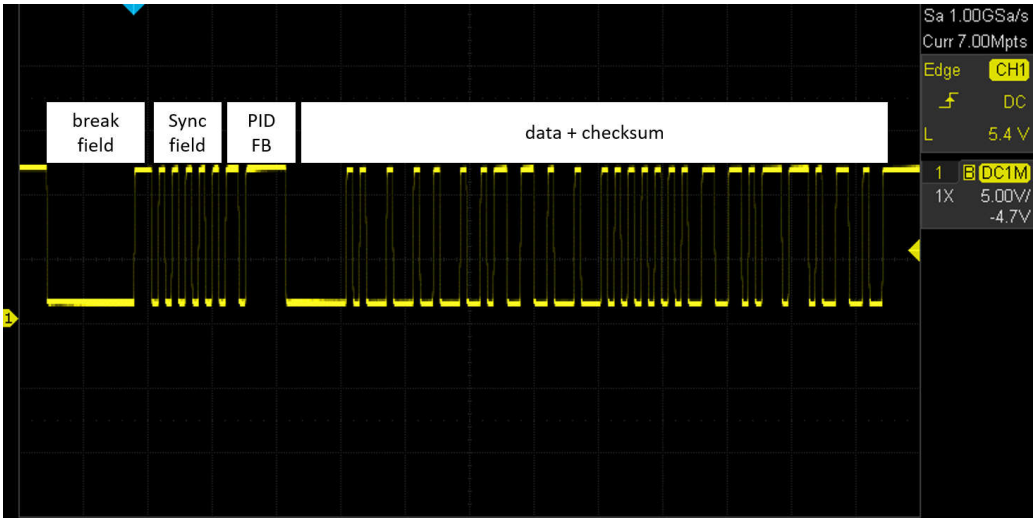

图 3-8. MCU 作为响应器接收数据的波形

gResponderRXBuffer	unsigned char[8]	[0x55 'U',0x55 'U',0x55 'U',0x55 'U',0x55 'U'...] (...
[0]	unsigned char	0x55 'U' (Hex)
[1]	unsigned char	0x55 'U' (Hex)
[2]	unsigned char	0x55 'U' (Hex)
[3]	unsigned char	0x55 'U' (Hex)
[4]	unsigned char	0x55 'U' (Hex)
[5]	unsigned char	0x01 '\x01' (Hex)
[6]	unsigned char	0x02 '\x02' (Hex)
[7]	unsigned char	0x03 '\x03' (Hex)

图 3-9. gResponderRXBuffer 结果

当命令器发送 0x08 时，结果如 图 3-10 和 图 3-11 中所示。主机设置为读取模式，并选择增强校验模式。然后，通信成功，主机成功接收到 MCU 发送的数据，可以通过波形确认。最后，GPIO 切换，以显示通信结束。

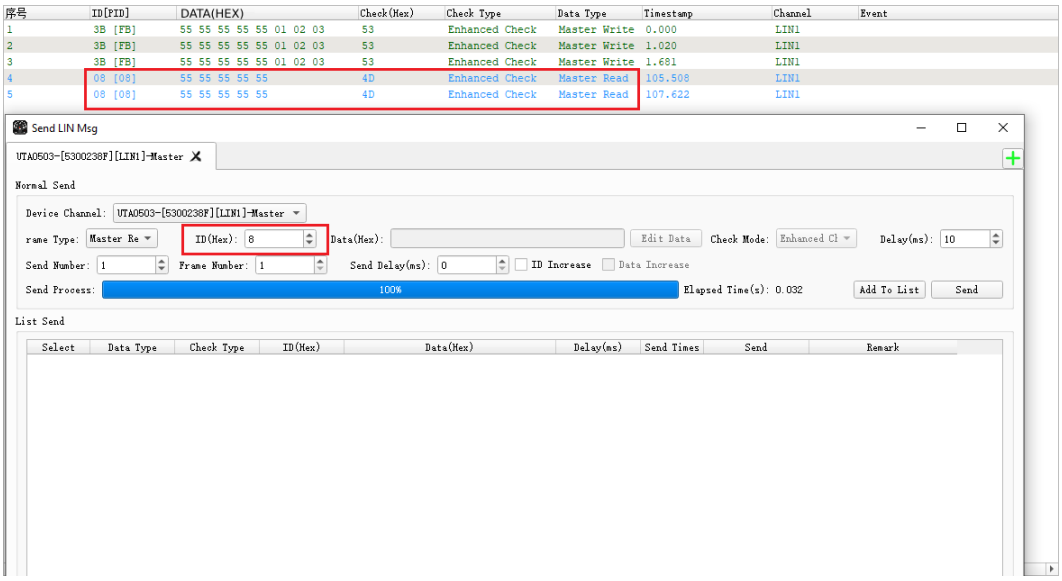

图 3-10. MCU 作为响应器传输数据的结果

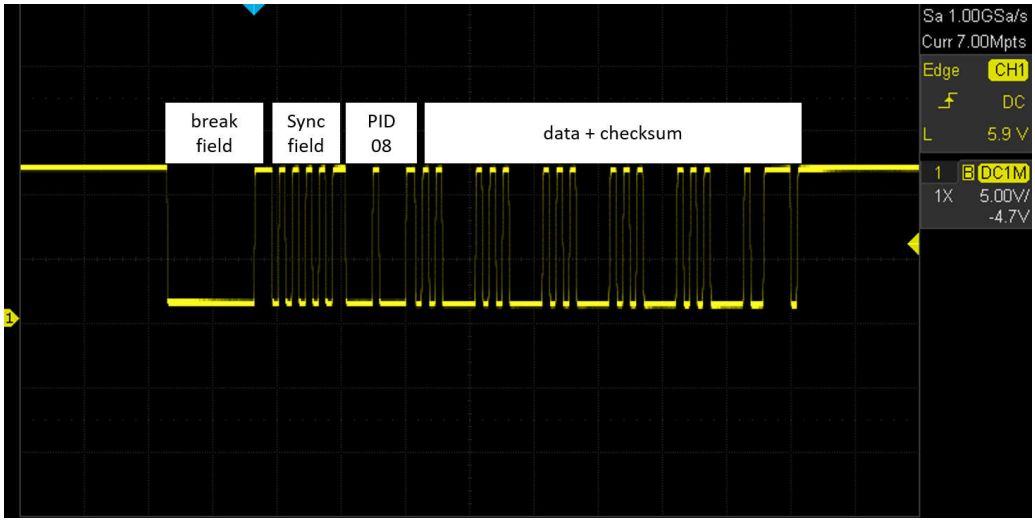

图 3-11. MCU 作为响应器传输数据的波形

4 总结

本文档简要介绍了 LIN 的基本知识，以及 MSPM0 如何通过硬件和软件支持 LIN 通信。本文档还提供了对 MSPM0 和 LIN 的初步了解，有助于加快开发进度。

5 参考资料

- 德州仪器 (TI)，*MSPM0 C 系列 24MHz 微控制器*技术参考手册。
- 德州仪器 (TI)，*MSPM0 L 系列 24MHz 微控制器*技术参考手册。
- 德州仪器 (TI)，*MSPM0 G 系列 24MHz 微控制器*技术参考手册。

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、某特定用途方面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功能安全、信息安全、监管或其他要求。

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。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/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。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265