

Subsystem Design

LIN 转 I2C 桥接器



1 说明

该子系统利用 MSPM0 器件充当 LIN 转 I2C 桥接器。当桥接器的 LIN 响应器从外部 LIN 命令器收集数据时，该过程开始；检测到有效消息后，数据将传输到桥接器的 I2C 控制器并发送到外部 I2C 目标器件。然后，目标器件使用新数据进行响应，桥接器随后会等待外部 LIN 命令器发出的请求，以便通过桥接器的 LIN 响应器传输捕获的数据。事务完成后，桥接器会将其状态复位为空闲。如果在任何阶段出现错误，I2C 和 LIN 接口都会表现出特定行为，以向用户发出故障信号。图 1-1 展示了用于 LIN 转 I2C 子系统的常见硬件配置。

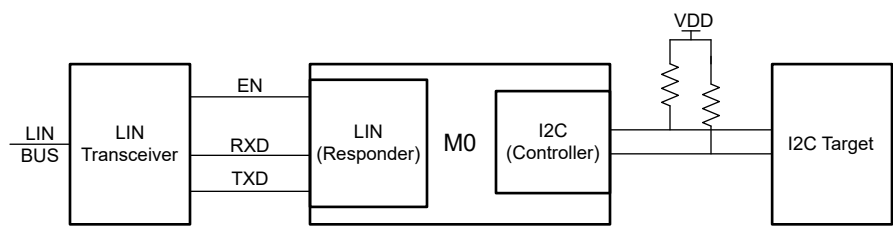


图 1-1. 系统功能方框图

2 所需外设

两个 MSPM0 模块用于该子系统：LIN 和 I2C。

表 2-1. MSPM0 模块

子块功能	外设使用	备注
LIN 响应器	LIN	在代码中称为 LIN_INST。
I2C 控制器	I2C	在代码中称为 I2C_INST。默认传输速率 400kHz。

3 兼容器件

该子系统与任何具有 UART-LIN 模块和 I2C 模块（无论是专用模块还是 UNICOMM 衍生模块）的 MSPM0 兼容。

4 设计步骤

- 在 [MSPM0 SDK](#)（MSP 子系统文件夹）中找到该工程，并将其导入到您的 CCS Studio 工作区中。
- 使用 lin_2_I2C.syscfg 为您的特定应用配置 I2C 和 LIN 模块。
- 设置最大数据包大小。注意：由于 LIN 每次传输限制为 8 个字节，因此如果传输更大的有效载荷，请验证您的实现是否考虑了多帧数据处理。

备注

由于 LIN 每次传输限制为 8 个字节，因此如果传输更大的有效载荷，请验证您的实现是否考虑了多帧数据处理。

5 设计注意事项

- 硬件选型

- a. 选择符合您特定应用要求（例如电压电平、数据速率和故障保护）的 LIN 收发器。德州仪器 (TI) 提供了广泛的 [LIN 收发器目录](#)，可用于确定适合您系统的理想器件。在本示例中，得益于易于遵循的用户指南，我们使用 [TLIN2029EVM](#) 作为 LIN 收发器。
 - b. I²C 引脚（以及 LIN 引脚，具体取决于系统要求）需要上拉电阻器才能正常运行。同样，如果任何 MSPM0 引脚配置为开漏（例如当前工程默认配置中的 PA0），则需要上拉电阻器，以使该引脚能够驱动其他元件或信号。
 - c. 查看系统内所有组件的电源、电压和电流要求。这包括验证每个模块的特定电气要求，例如 I²C、UART-LIN 和 GPIO 接口。
 - d. 评估 MSPM0 内部和外部振荡器的规格，以确保它们能为您的应用提供足够的时序精度。
2. 协议相关与固件逻辑
- a. 如果提高 I²C 速度，则需要根据 I²C 规范调整外部上拉电阻，以保持通信的完整性。一般而言，10k Ω 适用于 100kHz 工作频率。较高的 I²C 总线速率需要阻值较低的上拉电阻器；对于 400kHz 通信，必须使用更接近 4.7k Ω 的电阻器。
 - b. 可能需要进一步进行代码优化以适应更高的桥接器利用率。建议的优化包括提高器件运行速度、实现多个传输缓冲区、利用 DMA 或简化状态机。请注意，当前示例仅在 400kHz 的 I²C 速度下进行了测试。
 - c. 有关使用 MSPM0 实现 LIN 的更多建议，请参阅 [MSPM0 的 LIN 基础与实现指南](#)。

6 软件流程图

图 6-1 展示了子系统工程的流程图。

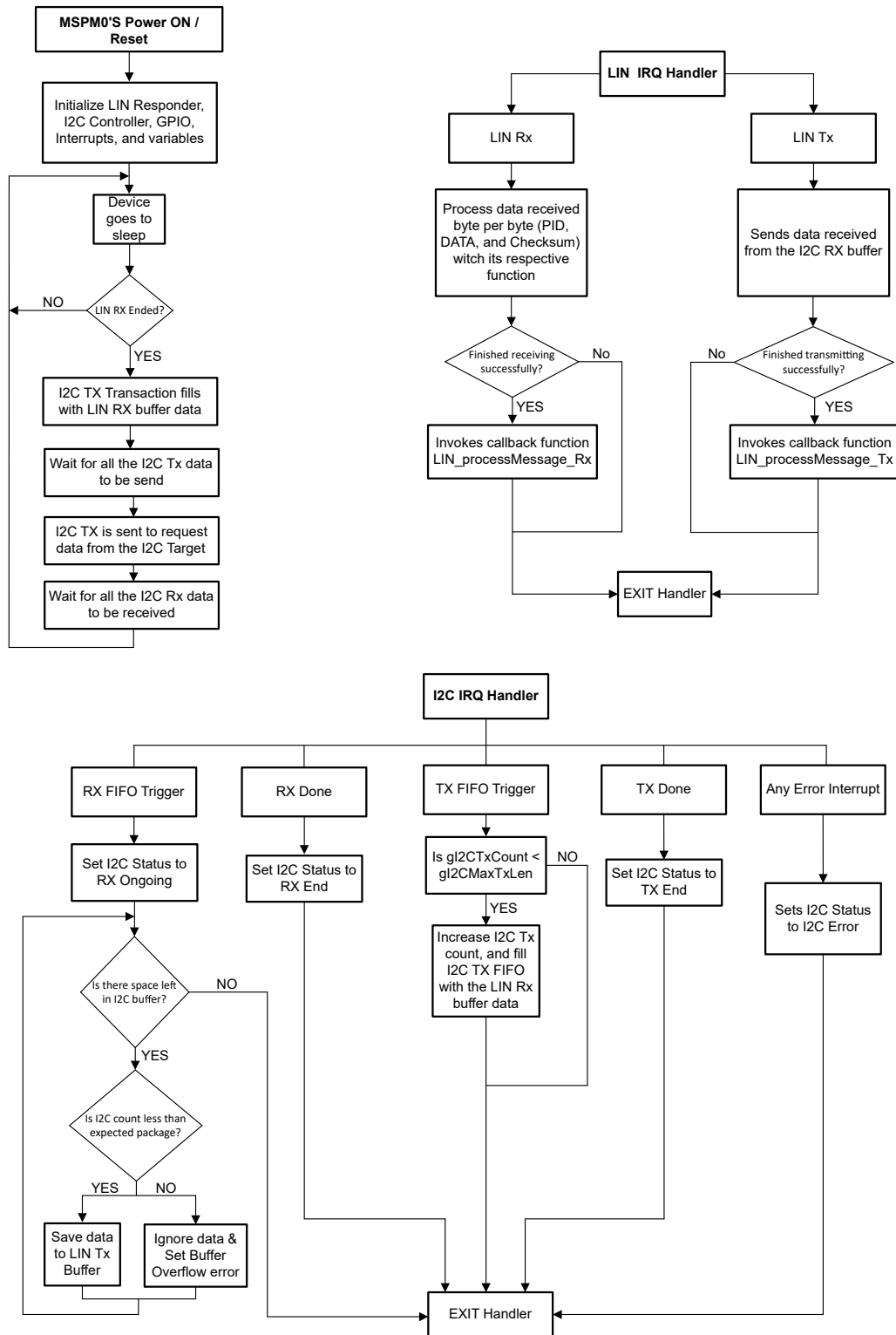


图 6-1. LIN 到 I2C 子系统软件流程图

图 6-2 展示了通信工作原理的高层时序图。X 字节表示外部 LIN 命令器发送的数据，而 Y 字节表示从 I²C 目标中检索并发送回 LIN 命令器的数据。

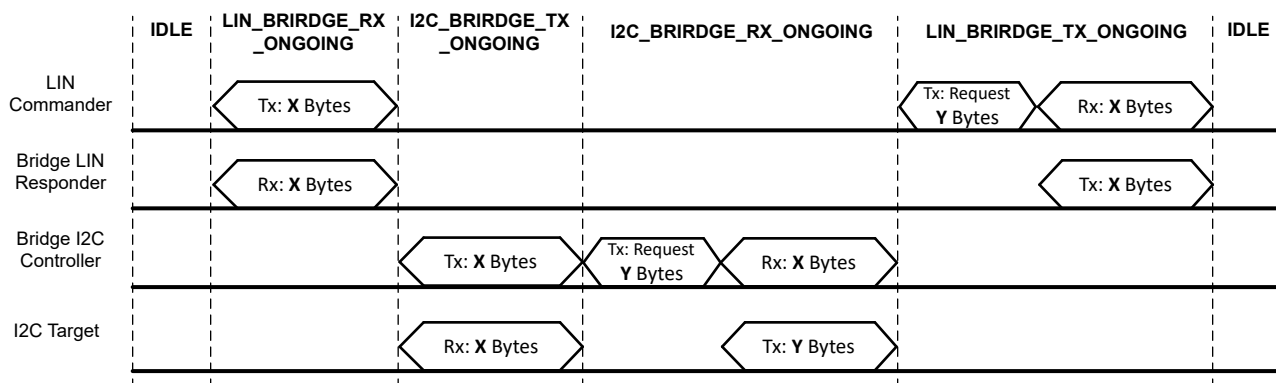


图 6-2. 高层通信图

7 应用代码

用于子系统的基本代码派生自随 MSPM0 SDK 提供的 `lin_responder` 和 `i2c_controller_rw_multibyte_fifo_interrupts` 示例。要更清楚地了解这两个示例，请参阅各自的 *README* 文件。

要展示桥接器结构，必须突出显示实现中每个组件的位置，以使用户知道在何处进行修改。

`lin_config.h` 文件是整个工程中使用的所有结构和全局变量的中央存储库，也是声明桥接器结构的位置，用于捕获桥接器在执行各个特定点所处的各个阶段。工程中的每个源文件都会引用一个标记为 `gBridgeState` 的全局变量，并随 LIN 或 I²C 事务各阶段的发生而更新。

```

/*! @enum BRIDGE_STATE - Bridge state machine states */
typedefenum {BRIDGE_IDLE = 0,
              BRIDGE_LIN_RX_STATE_ONGOING = 1,
              BRIDGE_LIN_RX_STATE_END = 2,
              BRIDGE_I2C_TX_STATE_ONGOING = 3,
              BRIDGE_I2C_RX_STATE_ONGOING = 4,
              BRIDGE_LIN_TX_STATE_ONGOING = 5,
            } BRIDGE_STATE;
volatile BRIDGE_STATE gBridgeState;

```

`lin_config.c` 文件在 MSPM0 上实现了 LIN 协议标准所需的机制。文件中的某些函数已进行修改，以跟踪 LIN 总线状态（例如 LIN_RX 和 LIN_TX 进行中的状态）。

转到主文件 `lin_to_i2c.c`，将调用 `lin_config.h` 以继承 LIN 和 I²C 模块中使用的所有定义和结构。之后，将创建以下项目来实现桥接器：

1. I²C 本地全局变量 `gTxLen/gTxCount` 和 `gRxLen/gRxCount`，用于跟踪 I²C 事务期间调度的字节数以及实际发送或接收的字节数。
2. 变量 `gl2cControllerStatus`，用于捕获 I²C 事务的每个阶段，包括 `IDLE`、`TX_STARTED`、`TX_INPROGRESS`、`TX_COMPLETE`、`RX_STARTED`、`RX_INPROGRESS`、`RX_COMPLETE` 以及 `ERROR`。此状态由 I²C ISR 更新，并由主循环轮询，以保持安全运行。

备注

在 I²C 流程中检测到的错误被视为 `arbitration-lost` 和 `NACK-related` 条件，并标记为相同。可以根据特定应用的需要添加其他错误状态和处理。

3. 与每个 PID 关联的 LIN 回调函数的声明（可用于在 LIN 事务的特定点采取行动；在本例中，这些回调在每个事务结束时被调用）。
4. `LIN_responderMessageTable` 表变量存储两种类型的消息：一个 `SUBSCRIBE` 条目，位于 ID `0x10`，其中响应器从命令器接收八个字节并执行 `LIN_processMessage_Rx` 回调；一个 `PUBLISH` 条目，位于 ID `0x20`，其

中响应器发送八个字节并执行 `LIN_processMessage_Tx` 回调 (在这两种情况下, 只有操作成功时才会调用回调)。

备注

为了最大限度地发挥桥接器的通用用途, 省略了某些原始的 LIN 响应器消息。有关全面的实现细节, 请参阅 MSPM0 SDK 中包含的 [lin_responder](#) 示例。

启动时, `SYSCFG_DL_init()` 会配置外设, 然后将 LIN-EN 分配的 I/O 引脚置为高电平以激活物理总线驱动器, 并执行一个循环, 根据各自的 ID 在 LIN 表中查找这两条消息的 PID 值。

然后, 表指针在全局 `gLIN` 变量中注册, 以便中断驱动的 LIN 驱动器知道要监控哪些 PID; 并在 LIN 计数器-比较寄存器中加载 `LIN_BREAK_FIELD_MIN_CYCLES`, 以建立最小的 BREAK 检测窗口。

最后, 清除桥接器所用模块 (UART-LIN、定时器和 I²C) 的挂起中断, 清除在各种桥接状态下切换的红色 LED, 并将 `gBridgeState` 和 `gl2cControllerStatus` 复位为其空闲值。

主循环以中断驱动的方式运行; CPU 执行 `__WFI()` 进入低功耗睡眠状态, 仅在外设生成中断时唤醒。

桥接器到达的第一个状态是 `BRIDGE_LIN_RX_STATE_ONGOING`, 当桥接器的 LIN 响应器接受来自外部 LIN 命令器的 `Subscribe` 消息时开始。成功接收 `X-bytes` 消息后, 将调用 `LIN_processMessage_Rx` 回调, 桥接器状态前进到 `BRIDGE_LIN_RX_STATE_END`, 红色 LED 切换一秒, 指示已达到结束状态。

此事件后返回主循环, I²C 序列开始。状态重置为 `BRIDGE_IDLE`, 如果接收到的 PID 与 `Subscribe` PID 匹配 (表明 LIN 消息已准备好进行 I²C 传输), 则将 LIN 接收缓冲区 (`gLIN.rxBuf`) 复制到 I²C 硬件 TX FIFO 中。如果这些字节可以在单次突发中传输完成, 则禁用 TXFIFO 触发中断; 否则中断保持启用状态, 以便 ISR 能在 FIFO 排空时予以补充。

随后, 系统将 I²C 状态标记为 `TX_STARTED`, 将桥接器状态设置为 `BRIDGE_I2C_TX_STATE_ONGOING`, 等待 I²C 总线变为空闲状态, 并调用 `DL_I2C_startControllerTransfer`, 发起将先前接收到的 LIN `X-bytes` 写入 I²C 配置中定义的地址的操作 (本例中为 `0x48`)。CPU 在循环中执行 `__WFI()`, 直到 ISR 发出 `TX_COMPLETE` 或 `ERROR` 信号; 任何错误都会通过 `__BKPT(0)` 触发软件断点。完成后, 红色 LED 再次切换, 以指示桥接器运行的 I²C-TX 部分已完成。

短暂延迟后, 系统将 I²C 状态标记为 `RX_STARTED`, 将桥接器状态设置为 `BRIDGE_I2C_RX_STATE_ONGOING`, 并从桥接器 I²C 控制器发出读取请求, 从外部 I²C 目标读取 `Y-bytes`。CPU 再次执行 `__WFI()`, 直到 ISR 发出 I²C 接收已完成的信号, 此时红色 LED 切换, 以指示桥接器运行的 I²C-RX 部分已完成。

最后, 外部 LIN 命令器请求从 I²C 目标中检索到的 `Y-bytes`。当桥接器的 LIN 响应器接受 `Publish` 消息时, 桥接器状态更改为最终状态 `BRIDGE_LIN_TX_STATE_ONGOING`。成功传输所有字节后, 将调用 `LIN_processMessage_Tx` 回调, 桥接器状态恢复为空闲, 红色 LED 切换一秒, 指示桥接器已完成其周期。此后, 桥接器状态和周期将重置为其初始值, 与程序开始时的状态一致。

如果出现错误, 预期会出现以下行为 (错误行为可以修改以适应应用的要求):

1. **LIN 错误**: LIN 全局变量内部错误值将设置为 `true`, 红色 LED 将闪烁 5 次。
2. **I2C 错误**: 代码将在软件断点处停止, I2C 状态全局变量将更新为错误。

8 结果

如 [LIN 至 I²C 事务捕获](#) 所示, 桥接器的运行过程如下:

1. 该周期由 LIN 命令器启动, 命令器将第一条消息传输到桥接器。
2. 桥接器通过 I²C 总线将此消息转发至 I²C 目标。
3. I²C 目标将请求的数据返回到桥接器。
4. 最后, LIN 命令器请求该数据, 桥接器提供响应。

在每个完整周期结束时, `RED_LED` 会切换 (低电平有效), 提供表明传输已成功处理的视觉指示。

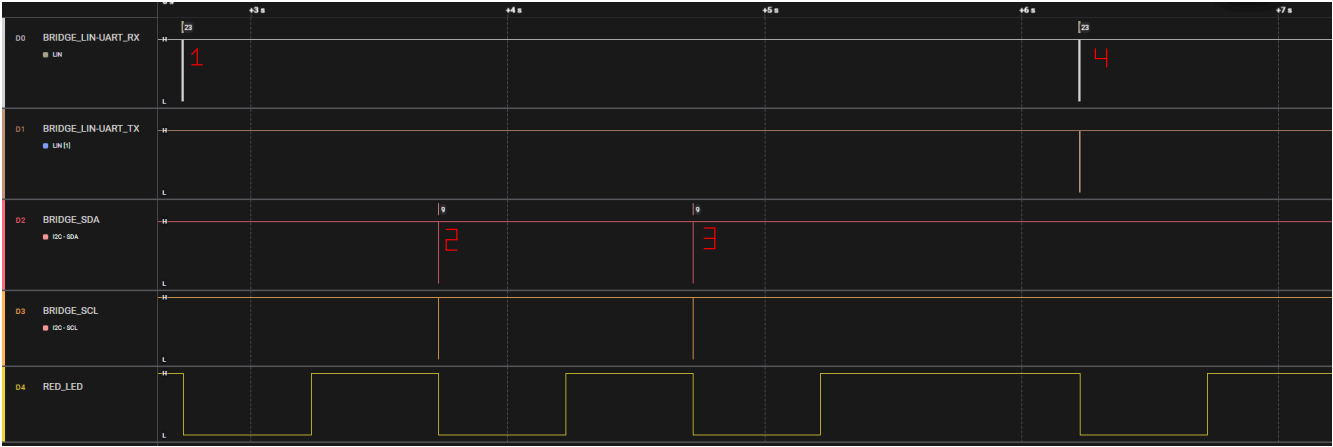


图 8-1. LIN 至 I²C 事务捕获

9 商标

所有商标均为其各自所有者的财产。

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。

版权所有 © 2026，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月