Subsystem Design

CAN 转 I2C 桥接器

Yuhao Zhao

设计说明

该子系统演示了如何构建 CAN-I2C 桥接器。CAN-I2C 桥接器使器件能够在一个接口上发送/接收信息，并在另一个接口上接收/发送信息 下载该示例的代码。这里提供了两个示例代码，以支持 I2C 分别在控制器模式或目标模式下工作。

图 1-1 显示了该子系统的功能图。请注意，这里为 IO 中断添加了一条线路，以实现从 I2C 目标到 I2C 控制器的消息传输。

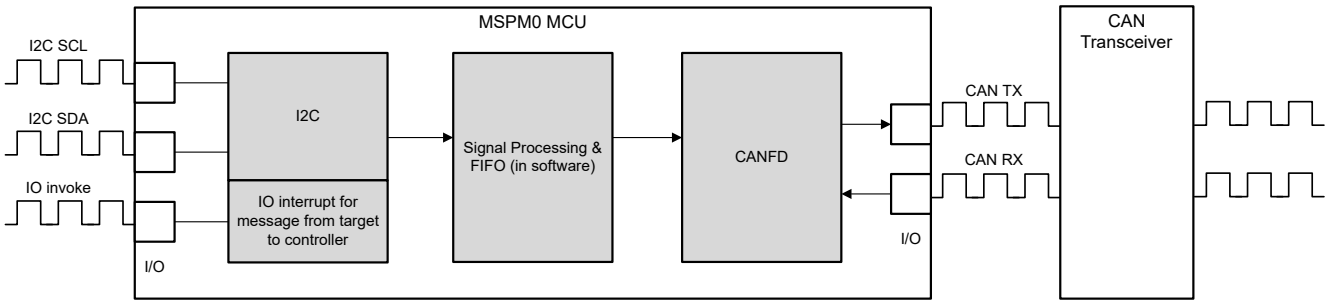

图 1-1. 子系统功能方框图

所需外设

此应用需要 CANFD 和 I2C。

表 1-1. 所需外设

子块功能	外设使用	说明
CAN 接口	(1x) CANFD	在代码中称为 *MCAN0_INST*
I2C 接口	(1 个) I2C	在代码中称为 *I2C_INST*

兼容器件

根据表 1-1 中的要求，该示例与表 1-2 中的器件兼容。相应的 EVM 可用于原型设计。

表 1-2. 兼容器件

兼容器件	EVM
MSPM0G35xx	LP-MSPM0G3507

设计步骤

1. 确定 CAN 接口的基本设置，包括 CAN 模式、位时序、消息 RAM 配置等。考虑应用中哪些设置是固定的，哪些设置已更改。在示例代码中，CANFD 的仲裁速率为 250kbit/s，数据速率为 2Mbit/s。
   a. CAN-FD 外设的主要特性包括：
      i. 具有 ECC 的专用 1KB 消息 SRAM

ii. 可配置的发送 FIFO、发送队列和事件 FIFO（最多 32 个元素）
iii. 多达 32 个专用发送缓冲器和 64 个专用接收缓冲器。两个可配置的接收 FIFO（每个 FIFO 最多 64 个元素）
iv. 多达 128 个滤波器元素

b. 如果启用 CANFD 模式：
i. 完全支持 64 字节 CAN-FD 帧
ii. 高达 8Mbit/s 比特率

c. 如果禁用 CANFD 模式：
i. 完全支持 8 字节传统 CAN 帧
ii. 高达 1Mbit/s 比特率

2. 确定 CAN 帧，包括数据长度、比特率切换、标识符和数据等。考虑应用中哪些部分是固定的，哪些部分需要更改。在示例代码中，标识符、数据长度和数据在不同帧中可能会发生变化，而其他项则固定不变。请注意，如果需要协议通信，用户需要修改代码。

```
/**
 *  @brief  Structure for MCAN Rx Buffer element.
 */
typedef struct {
    /*! Identifier */
    uint32_t id;
    /*! Remote Transmission Request
     *   0 = Received frame is a data frame
     *   1 = Received frame is a remote frame
     */
    uint32_t rtr;
    /*! Extended Identifier
     *   0 = 11-bit standard identifier
     *   1 = 29-bit extended identifier
     */
    uint32_t xtd;
    /*! Error State Indicator
     *   0 = Transmitting node is error active
     *   1 = Transmitting node is error passive
     */
    uint32_t esi;
    /*! Rx Timestamp */
    uint32_t rxts;
    /*! Data Length Code
     *   0-8  = CAN + CAN FD: received frame has 0-8 data bytes
     *   9-15 = CAN: received frame has 8 data bytes
     *   9-15 = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
     */
    uint32_t dlc;
    /*! Bit Rat Switching
     *   0 = Frame received without bit rate switching
     *   1 = Frame received with bit rate switching
     */
    uint32_t brs;
    /*! FD Format
     *   0 = Standard frame format
     *   1 = CAN FD frame format (new DLC-coding and CRC)
     */
    uint32_t fdf;
    /*! Filter Index */
    uint32_t fidx;
    /*! Accepted Non-matching Frame
     *   0 = Received frame matching filter index FIDX
     *   1 = Received frame did not match any Rx filter element
     */
    uint32_t anmf;
    /*! Data bytes.
     *   Only first dlc number of bytes are valid.
     */
    uint16_t data[DL_MCAN_MAX_PAYLOAD_BYTES];
} DL_MCAN_RxBufElement;
```

3. 确定 I2C 接口的基本设置，包括 I2C 模式、总线速度、目标地址、FIFO 等。考虑应用中哪些设置是固定的，哪些设置已更改。一个示例代码用于总线速度为 400kHz 的 I2C 控制器，另一个示例代码用于地址为 0x48 的 I2C 目标器件。

a. I2C 外设的主要特性包括：

i. 可配置为控制器或目标，比特率高达 1Mbps
ii. 用于接收和发送的独立 8 字节 FIFO
iii. 双目标地址功能，干扰抑制
iv. 针对 DMA 的独立控制器和目标中断生成以及硬件支持
v. 具有仲裁、时钟同步和多控制器支持的控制器运行

4. 确定 I2C 消息格式。通常，I2C 以字节为单位传输。为了实现高级别通信，用户可以通过软件实现帧通信。用户还可以根据需要引入特定的通信协议。在示例代码中，消息格式为 < 55 AA ID1 ID2 ID3 ID4 Length Data1 Data2 ...>。用户可以采用相同的格式通过 I2C 发送数据。55 AA 是标头。ID 区域为 4 字节。长度区域为 1 字节，表示数据长度。请注意，如果用户需要修改 I2C 数据包格式，则还需要修改帧采集和解析代码。

表 1-3. I2C 数据包格式

标头	地址	数据长度	数据
0x55 0xAA	4 字节	1 字节	(数据长度) 字节

5. 确定桥接器结构，包括需要转换哪些消息，如何转换消息等。
   a. 考虑桥接器是单向还是双向。通常每个接口都有两个功能：接收和发送。考虑是否只需要包含部分功能 (如 I2C 接收和 CAN 发送)。在示例代码中，CAN-I2C 桥接器是双向结构。由于 I2C 目标器件的接收和发送由 I2C 控制器控制，因此 I2C 目标器件无法启动到 I2C 控制器的传输。为了实现从目标到控制器的通信，该设计中增加了一条线路。目标器件的 IO 下拉会通知控制器要发送信息。
   b. 考虑要转换哪些信息以及相应的载体 (变量、FIFO)。在示例代码中，标识符、数据和数据长度从一个接口转换到另一接口。代码中定义了两个 FIFO，如图 1-2 所示。

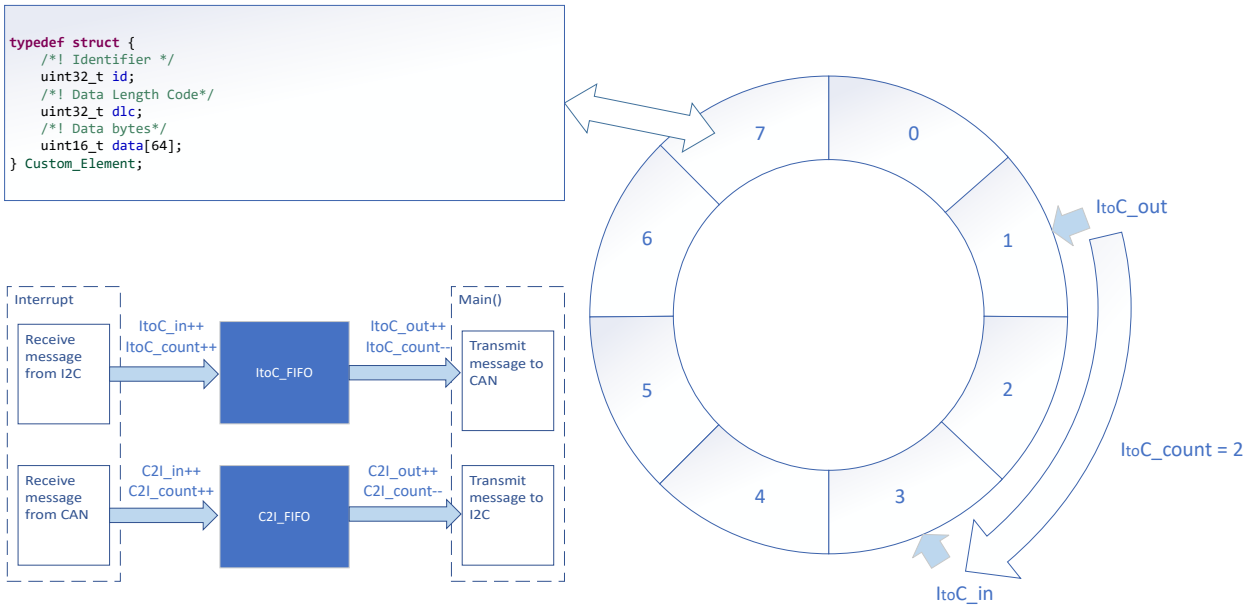

图 1-2. 桥接器结构

6. (可选) 考虑优先级设计、拥塞情况、错误处理等。

设计注意事项

1. 考虑应用中的信息流，确定各个接口需要接收或发送的信息和需要遵循的协议，并设计合适的信息传输载体来连接不同的接口。
2. 建议先单独测试接口，然后再实现整体桥接器功能。此外，还要考虑异常情况的处理，如通讯故障、过载、帧格式错误等。
3. 建议通过中断来实现接口功能，以确保及时通信。在示例代码中，接口功能通常在发生中断时实现，在 main() 函数中完成信息传递。

软件流程图

图 1-3 所示为 *CAN-I2C 桥接器* 的代码流程图，其中说明了如何在一个接口中接收消息并在另一个接口中发送消息。*CAN-I2C 桥接器* 可以分为四个独立的任务：从 I2C 接收、从 CAN 接收、通过 CAN 发送、通过 I2C 发送。两个 FIFO 实现双向消息传输和消息缓存。

请注意，I2C 是 I2C 控制器控制发送和接收的一种通信方法。通常，I2C 目标器件无法发起通信。对于 I2C 目标到控制器通信，I2C 目标器件可以在需要发送消息时下拉 IO，如 图 1-3 中所示。当检测到 IO 为低电平时，I2C 控制器可以在 IO 中断中启动 I2C 读取命令，如图 1-4 所示。在该演示中，可以将 I2C 配置为 I2C 目标或控制器。

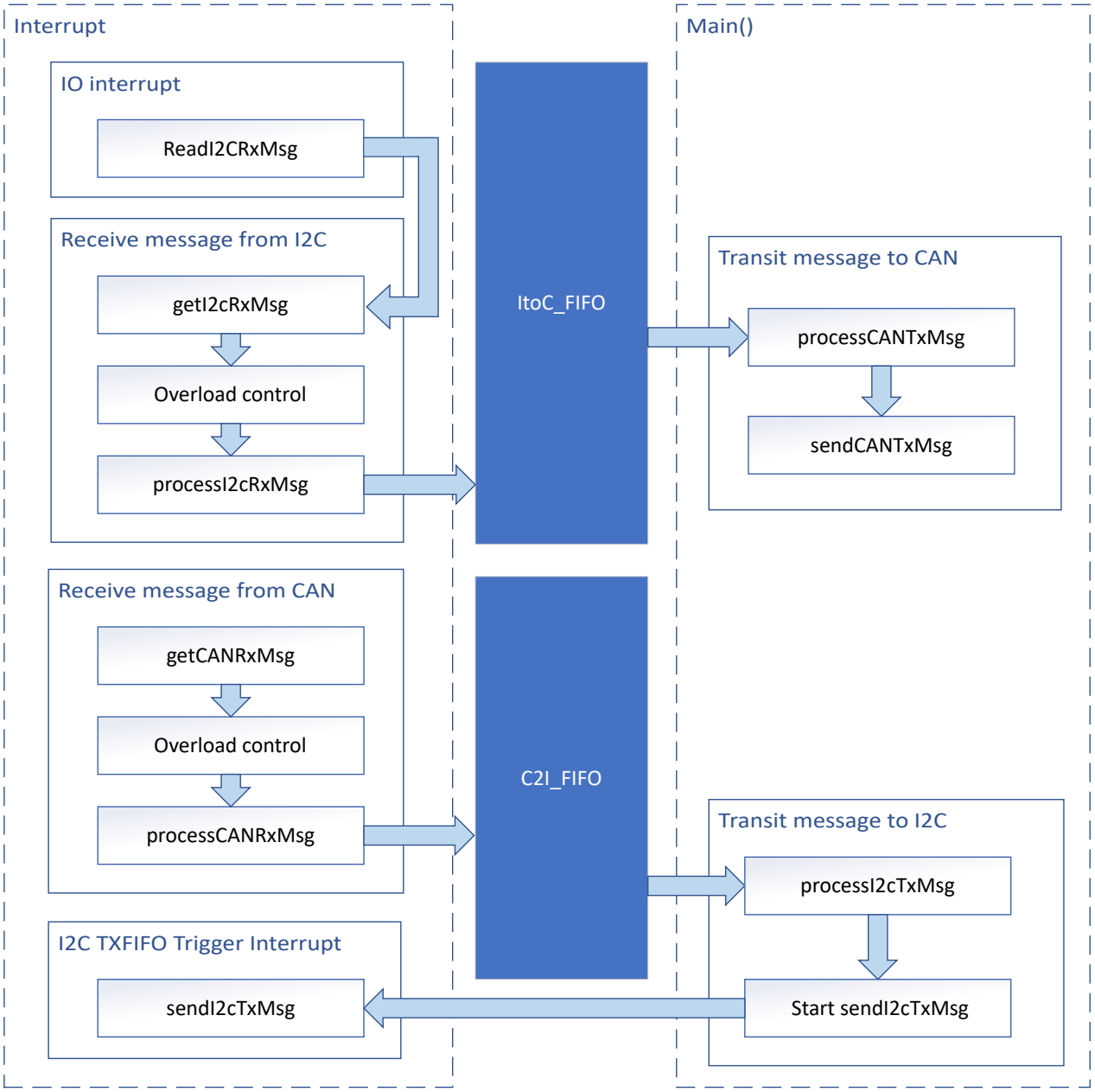

图 1-3. CAN-I2C（I2C 控制器）桥接器的应用软件流程图

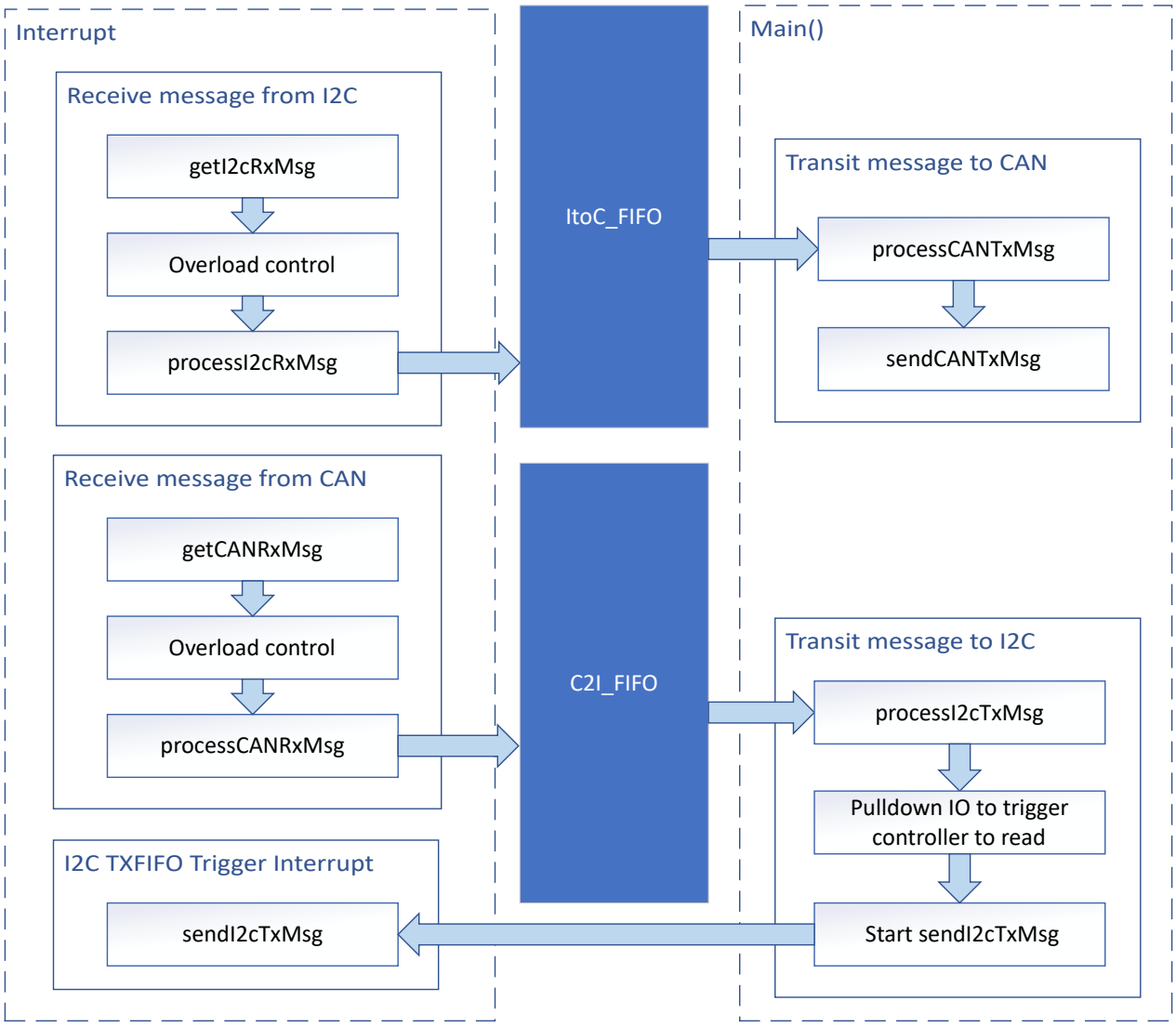

图 1-4. CAN-I2C（I2C 目标）桥接器的应用软件流程图

器件配置

该应用利用 TI 系统配置工具 (SysConfig) 图形界面为 CAN 和 I2C 生成配置代码。使用图形界面配置器件外设可简化应用原型设计过程。

图 1-3 中所述流程的代码可在图 1-5 所示的示例代码文件中找到。

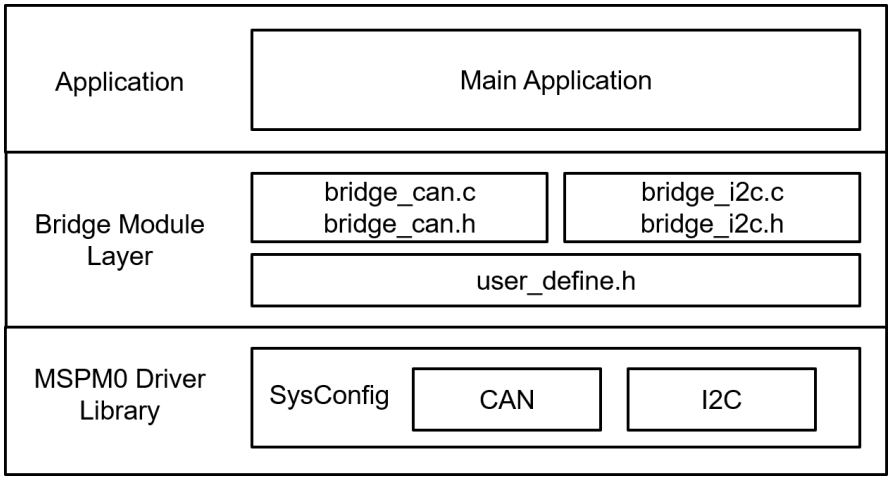

图 1-5. 文件结构

应用代码

以下代码片段显示了修改接口功能的位置。表中的函数被分类到不同的文件中。I2C 接收和发送函数包含在 bridge_i2c.c 和 bridge_i2c.h 中。CAN 接收和发送函数包含在 bridge_can.c 和 bridge_can.h 中。FIFO 元素结构在 user_define.h 中定义。

用户可以通过文件轻松分离函数。例如，如果只需要 I2C 函数，用户可以保留 bridge_i2c.c 和 bridge_i2c.h 以调用相应函数。

有关外设的基本配置，请参阅 MSPM0 SDK 和 DriverLib 文档。

表 1-4. 函数和说明

任务	函数	说明	位置
I2C 接收	readI2CRxMsg_controller()	向从器件发送读取请求（仅限 I2C 主器件）	bridge_i2c.c bridge_i2c.h
	getI2CRxMsg_controller()	获取接收到的 I2C 消息（仅限 I2C 主器件）	
	getI2CRxMsg_target()	获取接收到的 I2C 消息（仅限 I2C 从器件）	
	processI2cRxMsg()	转换接收到的 I2C 消息格式，并存储到 gI2C_RX_Element 中	
I2C 发送	processI2cTxMsg()	转换要通过 I2C 发送的 gI2C_TX_Element 格式	
	sendI2CTxMsg_controller()	通过 I2C 发送消息（仅限 I2C 主器件）	
	sendI2CTxMsg_target()	通过 I2C 发送消息（仅限 I2C 从器件）	
CAN 接收	getCANRxMsg()	获取接收到的 CAN 消息	bridge_can.c bridge_can.h
	processCANRxMsg()	转换接收到的 CAN 消息格式，并将消息存储到 gCAN_RX_Element 中	
CAN 发送	processCANTxMsg()	转换要通过 CAN 发送的 gCAN_TX_Element 格式	
	sendCANTxMsg()	通过 CAN 发送消息	

Custom_Element 结构在 user_define.h 中定义。Custom_Element 是 FIFO 元素、I2C/CAN 发送的输出元素和 I2C/CAN 接收的输入元素的结构。用户可以根据需要修改结构。

```
typedef struct {
    /*! Identifier */
```

```
    uint32_t id;
    /*! Data Length Code*/
    uint32_t dlc;
    /*! Data bytes*/
    uint16_t data[64];
} Custom_Element;
```

对于 FIFO，它有 2 个全局变量。有 6 个全局变量用于跟踪 FIFO。

```
Custom_Element ItoC_FIFO[ItoC_FIFO_SIZE];
Custom_Element C2I_FIFO[C2I_FIFO_SIZE];
uint16_t ItoC_in = 0;
uint16_t ItoC_out = 0;
uint16_t ItoC_count = 0;
uint16_t C2I_in = 0;
uint16_t C2I_out = 0;
uint16_t C2I_count = 0;
```

结果

通过使用 CAN 分析仪，用户可以在 CAN 侧发送和接收消息。作为演示，可以将两个 LaunchPad 用作两个 CAN-I2C 桥接器（一个 I2C 主器件和一个 I2C 从器件）以形成一个环路。当 CAN 分析仪通过主器件 LaunchPad 发送 CAN 消息时，它可以从从器件 LaunchPad 接收 CAN 消息。

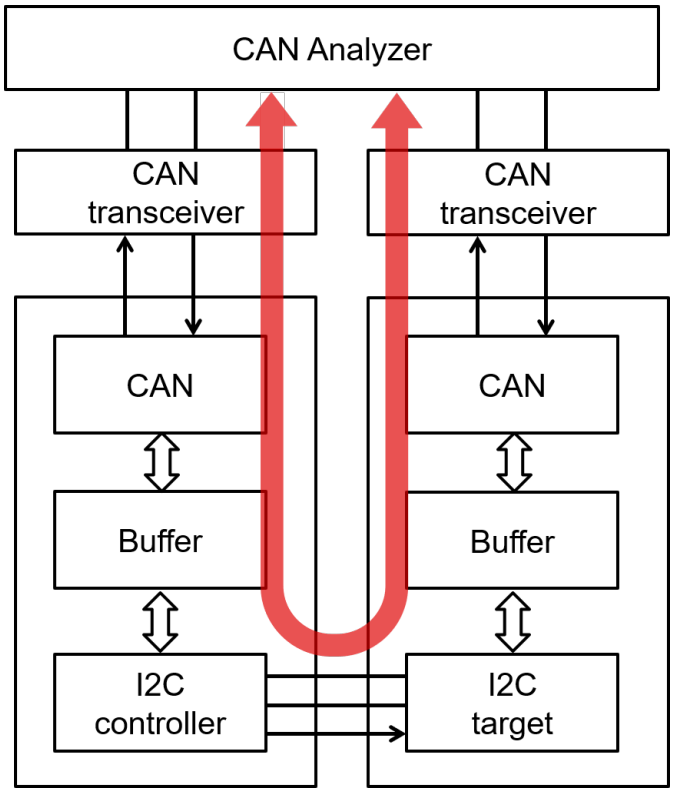

图 1-6. 演示

Index	Time	Device	Channel	Frame ID	Type	CANType	RT	Len	Data
					ALL	ALL	ALL		
0	0.000000	Device0	0	0x1	StandardFrame	CANFD Accelerate	Tx	16	00 11 22 33 44 53 66 77 88 99 AA BB CC DD EE FF
1	0.000900	Device0	1	0x1	StandardFrame	CANFD Accelerate	Rx	16	00 11 22 33 44 53 66 77 88 99 AA BB CC DD EE FF
2	75.392500	Device0	1	0x2	StandardFrame	CANFD Accelerate	Tx	16	00 11 22 33 44 53 66 77 88 99 AA BB CC DD EE FF
3	75.393400	Device0	0	0x2	StandardFrame	CANFD Accelerate	Rx	16	00 11 22 33 44 53 66 77 88 99 AA BB CC DD EE FF
4	96.807600	Device0	1	0x3	StandardFrame	CANFD Accelerate	Tx	12	00 11 22 33 44 53 66 77 88 99 AA BB
5	96.808400	Device0	0	0x3	StandardFrame	CANFD Accelerate	Rx	12	00 11 22 33 44 53 66 77 88 99 AA BB
6	111.433500	Device0	0	0x4	StandardFrame	CANFD Accelerate	Tx	8	00 11 22 33 44 53 66 77
7	111.434100	Device0	1	0x4	StandardFrame	CANFD Accelerate	Rx	8	00 11 22 33 44 53 66 77
8	127.068700	Device0	1	0x5	StandardFrame	CANFD Accelerate	Tx	4	00 11 22 33
9	127.069200	Device0	0	0x5	StandardFrame	CANFD Accelerate	Rx	4	00 11 22 33
10	137.580700	Device0	0	0x6	StandardFrame	CANFD Accelerate	Tx	4	00 11 22 33
11	137.581200	Device0	1	0x6	StandardFrame	CANFD Accelerate	Rx	4	00 11 22 33
12	160.259200	Device0	0	0x7	StandardFrame	CANFD Accelerate	Tx	1	00
13	160.259700	Device0	1	0x7	StandardFrame	CANFD Accelerate	Rx	1	00

图 1-7. CAN 分析仪针对演示发送和接收的消息

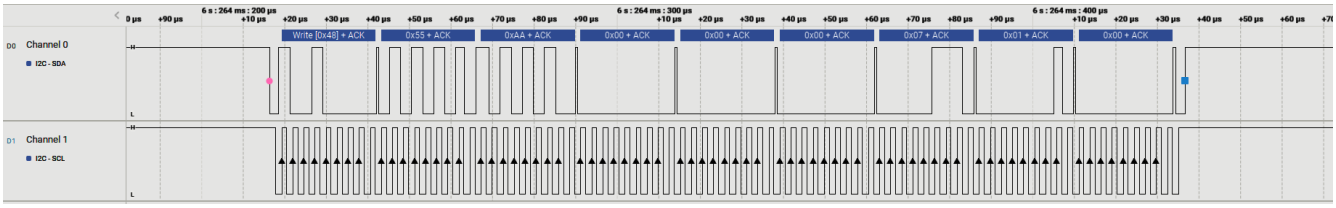

图 1-8. 逻辑分析仪的 PC 终端程序

其他资源

- 德州仪器 (TI)，下载 MSPM0 SDK
- 德州仪器 (TI)，详细了解 SysConfig
- 德州仪器 (TI)，MSPM0 G 系列 80MHz 微控制器技术参考手册
- 德州仪器 (TI)，MSPM0G LaunchPad 开发套件
- 德州仪器 (TI)，MSPM0 CAN Academy
- 德州仪器 (TI)，MSPM0 I2C Academy

重要声明和免责声明

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 产品的应用。严禁对这些资源进行其他复制或展示。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