AN12646

LPC546xx 系列中 USB CAN 的桥接

版本 0 — 2019 年 11 月

应用笔记

目录

1 简介

本文档描述了如何根据教程示例来构建实现 USB 总线和 CAN 总线之间的通信桥接。USB 选取 USB CDC 设备来传输数据。

该文档有助于：

1. 想熟悉并开始使用 USB CDC 应用程序；
2. 想了解 SDK CAN 驱动程序；
3. 想为他们的应用快速建立一个 USB_CAN 桥接。

1.1 术语

表 1. 术语表

项目	描述
CAN	控制器局域网
CDC	USB 通信设备类

2 实施

2.1 概述

本应用文档的目的是建立一个 USB_CAN 桥接，在这里 USB 数据重新传输到 CAN 总线，反之亦然。LPC54608 有两个 USB 控制器和两个 CAN 控制器。我们只使用一个 USB 控制器和一个 CAN 控制器。

表 2. 使用的 MCU 外设资源

使用的 IP	描述
USB1	使用 USB1 作为全速 USB 设备
CAN0	CAN0 接口

系统框图如 图 1 所示。SDK 已经提供了 CAN 驱动程序和 USB 堆栈。您需要为每个通道添加两个缓冲区，一个用于 USB->CAN 总线，另一个用于 CAN ->USB 总线。两个通道相互独立以确保最佳的性能。

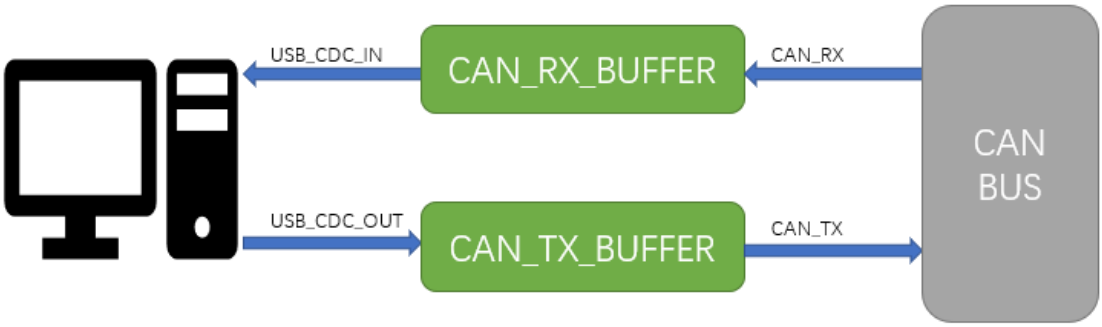

图 1. 系统框图

2.2 相关 SDK 示例

该软件基于两个 SDK 示例：

2.2.1 MCAN 环回示例

MCAN 是一个简单的 CAN 环回示例，演示了 LPC54608 的 CAN 模块的用法。此示例启用 CAN 模块的内部环回并发送 CAN 帧，CAN 帧环回到 CAN 接收器，MCU 在 UART 终端上显示任何接收到的 CAN 帧。我们建议用户阅读示例介绍文档，并运行示例程序以熟悉这个示例。

示例程序位置：

```
\SDK_2.6.0_LPC54608J512\boards\lpcxpresso54608\driver_examples\mcan\loopback
```

2.2.2 usb_device_cdc_vcom 示例

USB CDC 类示例将 USB 作为为通信设备。设置完成后，设备上会弹出一个 COM 端口，任何字符都会通过这个 COM 端口环回显示。有关如何安装设备驱动程序和运行示例，请参阅本示例介绍文档。

示例位置：

```
SDK_2.6.0_LPC54608J512\boards\lpcxpresso54608\usb_examples\usb_device_cdc_vcom\bm
```

2.3 将这两个例子结合成 USB CAN 桥接应用

以上两个 SDK 示例提供了实现构建 USB-CAN 桥接的模块。软件需要执行以下任务：

1. 将 CAN 驱动程序集成到 usb_device_cdc_vcom 示例中，并将 CAN0 模块启用和配置为环回模式。
2. USB 和 CAN 以不同的速度运行，我们需要配置两个 FIFO，一个用于 CAN Tx 缓冲区，一个用于 CAN Rx 缓冲区，如图 1 所示。
3. 连接这些通信接口，当 USB 输出数据包到达时，将数据缓冲到 CAN 发送缓冲区。当 CAN 总线接收到一个数据帧时，将数据缓冲到 CAN\U RX 缓冲区。
4. 创建两个单独的线程，一个用于 CAN 发送：当 CAN 发送缓冲区不为空时，获取缓冲区数据并发送到 CAN 总线。另一个线程服务于 USB 接口：当 CAN 接口缓冲区不为空时，获取缓冲区数据并通过 USB CDC 发送到 PC。

主要配置方式如下：

1. 在 CAN0 中断处理程序中，当接收到 CAN 数据帧时，将数据存放到 CAN\U RX\U 缓冲区：

```
        void CAN0_IRQ0_IRQHandler(void)
        {
        static mcan_rx_buffer_frame_t rxFrame;
```

```
        MCAN_ClearStatusFlag(CAN0, CAN_IR_RF0N_MASK);
        MCAN_ReadRxFifo(CAN0, 0, &rxFrame);
        msg_t msg;
        msg.cmd = MSG_CAN_RX;
        msg.len = rxFrame.dlc;
        memcpy(msg.buf, rxFrame.data, msg.len);
        /* push data into CAN_RX_BUFFER */
        mq_push(msg);
        }
```

2. 在 APPTask 函数中，轮询消息缓冲区，查看是否有消息进入，如果接收到 CAN_TX 消息，则调用 SDK 驱动程序 CAN_TX 函数向 CAN 总线发送数据。如果接收到 CAN_RX 消息，调用 USB Stack API: USB_DeviceCdcAcmSend 将数据发送到 PC。

```
        void APPTask(void)
        {
        msg_t can_tx_msg;
        msg_t *pMsg;
        uint8_t usb_tx_flag = 0;
        if ((1 == s_cdcVcom.attach) && (1 == s_cdcVcom.startTransactions))
        {
        if ((0 != s_recvSize) && (0xFFFFFFFFU != s_recvSize))
        {
        /* push data to CAN_TX_BUFFER */
        can_tx_msg.cmd = 0;
        can_tx_msg.len = s_recvSize;
        memcpy(can_tx_msg.buf, s_currRecvBuf, can_tx_msg.len);
        mq_push(can_tx_msg);
        s_recvSize = 0;
        }
        /* handle messages */
        if(mq_exist())
        {
        pMsg = mq_pop();
        switch(pMsg->cmd)
        {
        case MSG_USB_RX:
        app_can_send(CAN_TX_ID, pMsg->buf, pMsg->len)
        break;
        case MSG_CAN_RX:
        //dump_data(pMsg->buf, pMsg->len);
        memcpy(s_currSendBuf, pMsg->buf, pMsg->len);
        USB_DeviceCdcAcmSend(s_cdcVcom.cdcAcmHandle, USB_CDC_VCOM_BULK_IN_ENDPOINT,
s_currSendBuf, pMsg->len);
        usb_tx_flag = 1;
        break;
        }
        }
        /* send a empty USB_IN packet */
        if(usb_tx_flag == 0)
        {
        USB_DeviceCdcAcmSend(s_cdcVcom.cdcAcmHandle, USB_CDC_VCOM_BULK_IN_ENDPOINT,
s_currSendBuf, 0);
        }
        usb_tx_flag = 0;
        }
        }
```

3 测试

由于启用了 CAN 环回模式，并且 CAN Tx 帧 ID 与 CAN Rx FIFO 相同，因此任何由 CAN0 发送的数据可以环回发送回自身。因此，任何数据从 PC 发送的数据可以环回发送回 PC。我们使用这一点来检查代码的功能。

将 USB 电缆插入 J3 USB 全速端口，界面弹出显示 USB CDC 设备，如 图 2 所示：

图 2. USB CDC COM 端口

使用 Putty 在 open COM 端口上输入小于等于 8 字节的字符，数据环回显示在终端。如 图 3 所示：

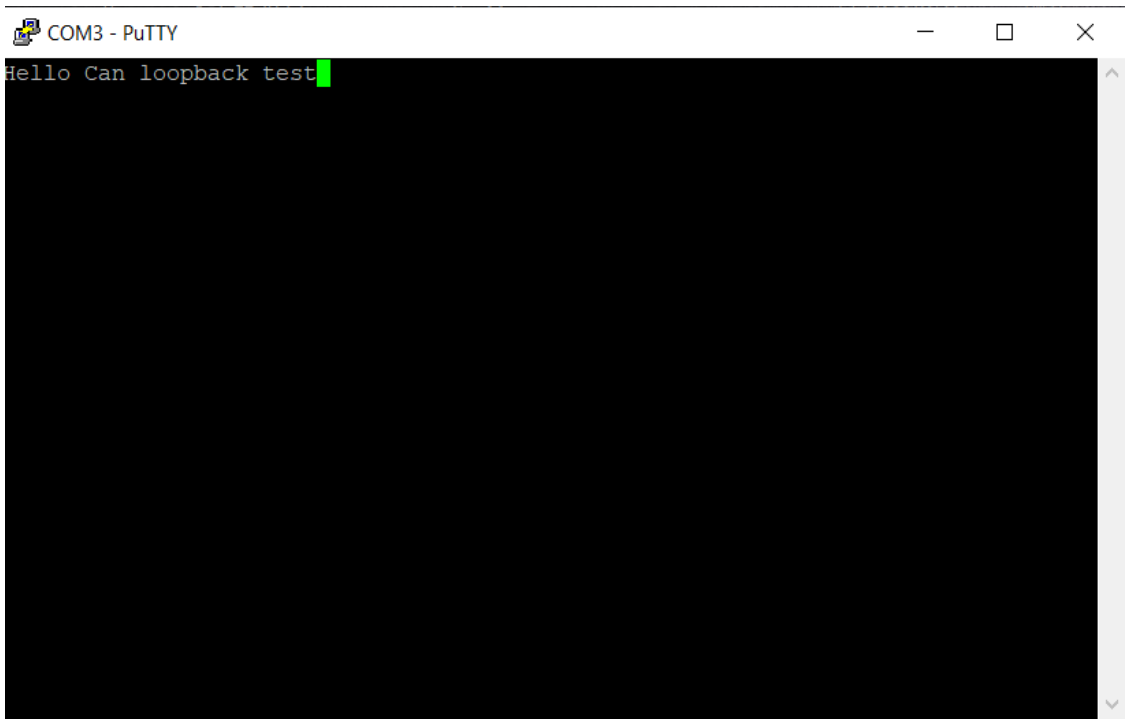

图 3. SB_CAN 环回测试

4 结论与局限

本文介绍了一种基于 MCUxpresso SDK 构建 CAN_USB 桥接的简单方法，并给出了使用异步通信软件通道的一个通用软件架构。

本示例中的限制包括：

1. 在此示例中，CAN Tx 帧和 CAN Rx 帧都使用 ID 为 0x123 的标准数据帧。
2. 根据 CAN2.0 规范，一个 CAN 数据帧最多只能承载 8 字节的数据。因此，USB CDC 一次最多只能发送 8 个字节。如果数据长度大于 8 字节，则丢弃超出的数据。

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2019 年 11 月

Document identifier: AN12646

arm