AN12646

USB CAN Bridge for LPC546xx

Rev. 0 — November 2019 Application Note

Contents

1 Introduction

This document describes how to build a tutorial like example to implement a communication bridge between USB bus and CAN-bus. USB is enumerated as a USB CDC device to transmit data.

The document is helpful to:

1. Anyone who wants to familiar and getting started with USB CDC application.
2. Anyone who wants to get knowledge for SDK CAN driver usage.
3. Anyone who wants to build a USB_CAN bridge quickly for their application usage.

1.1 Glossary

Table 1. Abbreviation

Items	Description
CAN	Controller area network
CDC	USB communication device class

2 Implementation

2.1 Overview

The aim of this application note is to build a USB_CAN bridge where the USB data retransmit to CAN-bus and vice versa. The LPC54608 has 2 USB controllers and 2 CAN controllers. We only use one USB controllers and one CAN controller.

Table 2. MCU peripheral resource used

IP used	Description
USB1	Use USB1 as full speed USB device
CAN0	CAN0 interface

System block diagram is shown in Figure 1 .The CAN driver and USB Stack is already provided by SDK. You need to add 2 buffers for each pipe, one for USB->CAN-bus, another for CAN-bus->US. The two pipes are independent to make sure best performance.

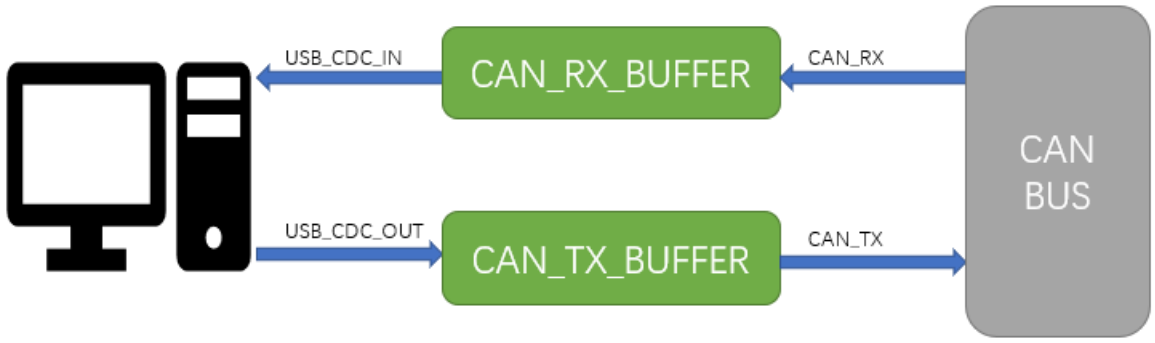

Figure 1. System block diagram

2.2 Related SDK example

The software is based on two SDK examples:

2.2.1 MCAN loopback example

MCAN is a simple CAN loopback example which demonstrates usage of LPC54608's CAN module. This example enables CAN module's internal loopback and send a CAN frame, the CAN frame loopbacks into CAN receiver and MCU displays any received CAN frames on UART terminal. We suggest user read the readme documentation, and running it in order to get familiar with this example.

Example location:

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

2.2.2 usb_device_cdc_vcom example

USB CDC class example enumerates USB as a communication device class. When USB enumerates complete, a COM port pops out on device, any character send through this COM port loopback to display. See readme documentation for this example for how to install device driver and run the demo.

Example location:

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

2.3 Combine the two examples into USB CAN bridge application

The above two SDK examples provide the building block for implement a USB-CAN bridge. The software need to do the following tasks:

1. Integrate CAN driver into usb_device_cdc_vcom example and enable and configure CAN0 module into loopback mode.
2. USB and CAN are running with different speed, we need to implement two FIFOs, one for CAN Tx buffer and one for CAN Rx buffer as Figure 1 shows
3. Link those communication interfaces, when USB_OUT packet arrived, buffer data into CAN_TX buffer. When CAN-bus receive a frame, buffer data into CAN_RX buffer.
4. Create two separate threads, one use for CAN_TX send: when CAN_TX buffer is not empty, fetch buffer data and send to CAN-bus. Another thread serves USB_IN: when CAN_RX buffer is not empty, fetch buffer data and send to PC via USB CDC.

.

The main modifications are list as follows:

1. In CAN0 interrupt handler, when a CAN frame received, buffer data into CAN_RX_BUFFER:

```
        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. In APPTask function, poll message buffer to see if any message come in, if there is CAN_TX message received, call SDK driver CAN_Tx function to send data to CAN-bus. If there is CAN_RX message received, call USB Stack API: **USB_DeviceCdcAcmSend** to send data to 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 Test

Since CAN loopback mode is enabled and CAN Tx frame ID is same with CAN Rx FIFO, any data send by CAN0 loopback to itself. Therefore, any data send from PC loopback to PC. We use this point to check the functionality of the code.

Plug USB cable to J3 USB full speed port, the USB CDC device pops up as Figure 2 shows:

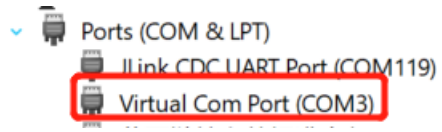

Figure 2. USB CDC COM port

Using Putty on open COM port, input any character less or equal than 8 bytes, the data loopback on terminal. See Figure 3shows:

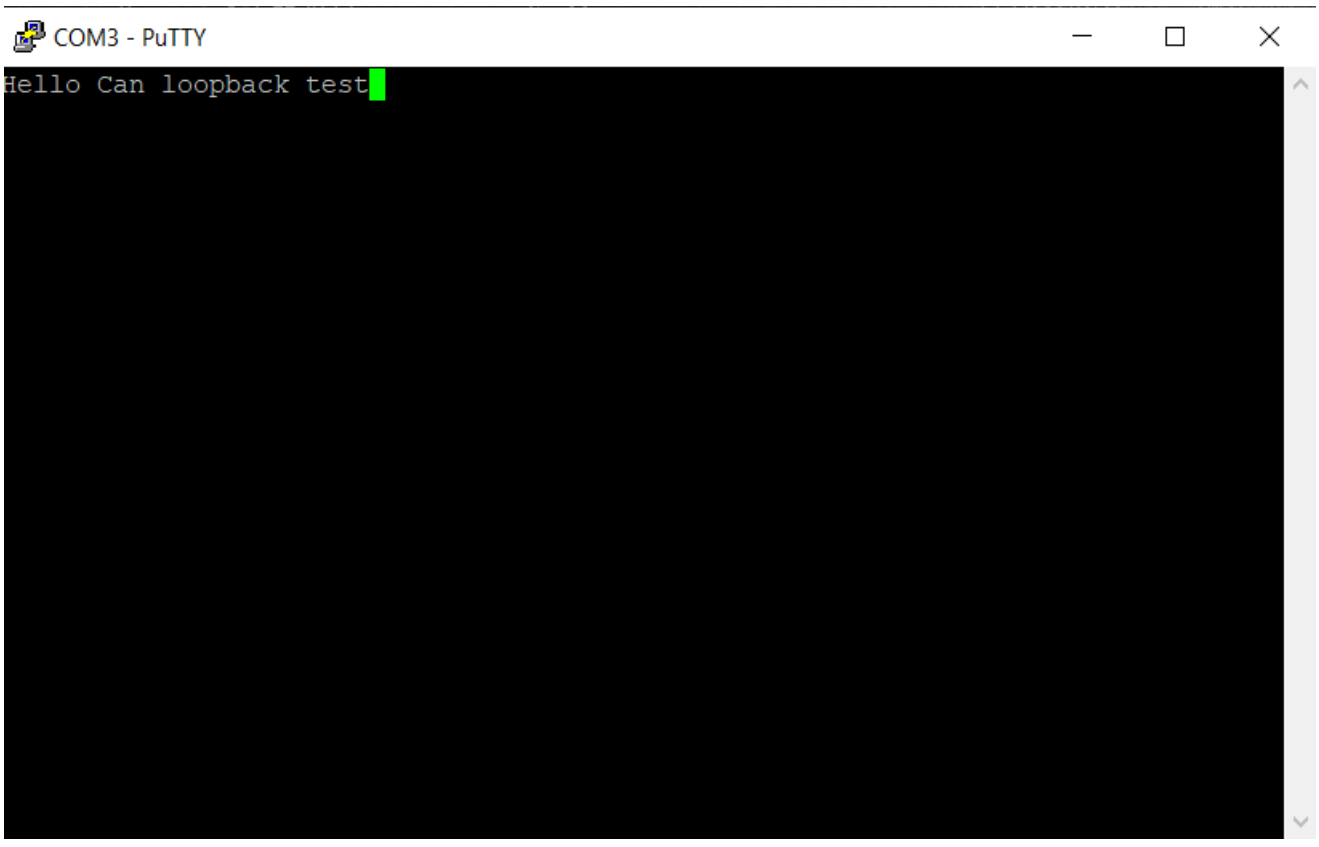

Figure 3. SB_CAN loopback test

4 Conclusion and limitation

This document explains a simple way to build a CAN_UART bridge base on MCUXpresso SDK software and provides a general architecture for async communication software pipes.

The limitations are:

1. In this example, CAN Tx frame and CAN Rx frame are all using standard data frame with ID:0x123
2. According to CAN2.0 specification, one CAN data frame can only carry up to 8 bytes data. So, USB CDC can only send up to 8 bytes at one time. If data length is larger than 8 bytes, the remainings are discarded.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

Date of release: November 2019

Document identifier: AN12646

arm