AN13515

基于LPC55S16的USB CAN适配器

第1版 — 2022年1月18日 应用笔记

目录

1 介绍

1.1 概述

本应用笔记旨在构建一个USB-CAN适配器，USB数据可以通过该适配器再传输到CAN总线，反之亦然。NXP LPC55S16有一个高速USB端口和CANFD控制器。HSUSB可以达到480 Mbit/s的传输速率，足以在最高的CAN波特率下传输CANFD帧。

为了让系统易于使用并能与其他设备兼容，采用USB CDC虚拟COM端口作为与PC通讯的接口，并使用一个从开源项目USBtinViewer延续下来的简单的ASCII协议。

2 实施方案

2.1 概述

如图1所示，USB CDC使用两个USB物理批量传输端点以在PC和MCU之间传输数据。每个端点负责单向的数据传输。

SDK已经提供了MCAN驱动程序和USB协议栈。在软件中，为每个管道添加了两个缓冲区，一个用于USB->CAN总线，另一个用于CAN总线->USB。为了保证最佳的性能，这两个管道是独立的。

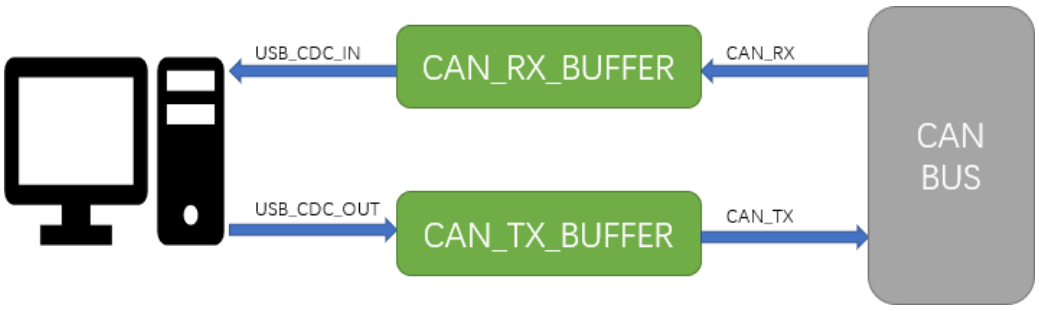

图1. 系统框图

2.2 相关的SDK示例

在继续任务之前，需要了解USB CDC和CAN的用法的背景知识。幸而，SDK提供了两个示例。

- MCAN的环回示例

MCAN示例是一个简单的CAN环回示例，演示了LPC54608的CAN模块的用法。此示例启用了CAN模块的内部环回并发送一个CAN帧。该CAN帧环回到CAN接收器，之后MCU在UART终端上显示所有接收到的CAN帧。要熟悉此示例，请阅读readme文档并运行该示例。

示例位置：*\boards\lpcxpresso55s16\driver_examples\mcan\loopback*

- usb_device_cdc_vcom示例

  此示例是USB CDC类的示例，将USB枚举为通信设备类。当USB的枚举完成后，器件上会弹出一个COM端口。通过此COM端口发送的所有字符都会环回显示出来。有关如何安装器件驱动程序和如何运行此示例的信息，请参见此示例的readme文档。

  示例位置：*\boards\lpcxpresso55s16\usb_examples\usb_device_cdc_vcom\bm*

在继续阅读此文档之前，请先熟悉以上两个示例。这两个示例是USB-CAN适配器设计的基石。

2.3 硬件

表1列出了USB-CAN适配器中使用的GPIO引脚。

表1. USB-CAN适配器中使用的GPIO引脚

GPIO	信号	说明
PIO1_2	CAN0_TX	CAN总线信号
PIO1_3	CAN0_RX	CAN总线信号
USB1_DM	USB1_DM	HSUSB DM
USB1_DP	USB1_DP	HSUSB DP
PIO0_29	UART_RXD	调试UART RXD
PIO0_30	UART_TXD	调试UART TXD

有关完整的原理图，请参见“USB CAN适配器的原理图”。

2.4 串行通信协议

USB-CAN适配器会在主机上注册为一个虚拟的串行端口。使用简单的ASCII命令，可以通过该串行端口控制CAN总线的配置。用户可以从任何串行终端程序或自己的程序发送/接收命令。

表2. ASCII协议的命令列表：

ASCII命令	响应	说明
O[CR]	[CR]	打开CAN通道
C[CR]	[CR]	关闭CAN通道
tiiildd..[CR]	发送标准（11位）帧。	iii：十六进制格式的标识符（000-7FF） l：数据长度（0-8） dd：十六进制格式的数据字节值（00-FF）

表格续下一页...

表2. ASCII协议的命令列表：（续）

ASCII命令	响应	说明
Sx[CR]	设置波特率	x：比特率id（0-8） S0 = 10 kBaud S1 = 20 kBaud S2 = 50 kBaud S3 = 100 kBaud S4 = 125 kBaud S5 = 250 kBaud S6 = 500 kBaud S7 = 800 kBaud S8 = 1 MBaud
Tiiiiiiiildd..[CR]	发送扩展（29位）帧。	iiiiiiii：十六进制格式的标识符（0000000-1FFFFFFF） l：数据长度（0-8） dd：十六进制格式的数据字节值（00-FF）
riiil[CR]	发送标准RTR（11位）帧。	iii：十六进制格式的标识符（000-7FF） l：数据长度（0-8）
Riiiiiiiil[CR]	发送扩展RTR（29位）帧。	iiiiiiii：十六进制格式的标识符符（0000000-1FFFFFFF） l：数据长度（0-8）
mxxxxxxxx[CR	设置接收过滤器掩码	SJA1000格式（AM0……AM3）。 只有前11位是相关的。 xxxxxxxx：接收过滤器掩码
Mxxxxxxxx[CR]	设置接收过滤器代码。	SJA1000格式（AC0……AC3）。 只有前11位是相关的。 xxxxxxxx：接收过滤器代码。

示例：

设置10 k波特率，打开CAN通道，发送CAN消息（id=001 h，dlc=4，data=11 22 33 44），然后关闭CAN。

表3. CAN消息

命令	响应
S0[CR]	[CR]
O[CR]	[CR]

表格续下一页...

表3. CAN消息（续）

命令	响应
t001411223344[CR]	z[CR]
C[CR]	[CR]

通过状态机，软件会接收来自CDC端口的串行流，解析ASCII码，并应用此命令。下面列出了该软件中使用的一些重要代码片段。有关完整的源代码，请参见AN13515SW。

- 要发送一个CAN帧，

```
txFrame.xtd = kMCAN_FrameIDStandard;
txFrame.rtr = kMCAN_FrameTypeData;
txFrame.fdf = 0;
txFrame.brs = 0;
txFrame.dlc  = len;
txFrame.id   = id << STDID_OFFSET;
txFrame.data = buf;
txFrame.size = CAN_DATASIZE;

txXfer.frame     = &txFrame;
txXfer.bufferIdx = 0;
MCAN_TransferSendNonBlocking(EXAMPLE_MCAN, &mcanHandle, &txXfer);
```

- 要接收一个CAN帧，

```
static void mcan_callback(CAN_Type *base, mcan_handle_t *handle, status_t status, uint32_t
result, void *userData)
{
   switch (status)
   {
      case kStatus_MCAN_RxFifo0Idle:
      {
         memcpy(rx_data, rxFrame.data, rxFrame.size);
         MCAN_TransferReceiveFifoNonBlocking(EXAMPLE_MCAN, 0, &mcanHandle, &rxXfer);
         can_rx_cb(rxFrame.id >> STDID_OFFSET, rx_data, rxFrame.dlc);
       }
       break;

       case kStatus_MCAN_TxIdle:
       {

        }
                    break;

        default:
           break;
   }
}
```

- 要通过USB CDC发送数据，

```
void usbd_cdc_send(uint8_t *buf, uint32_t len)
{
    USB_DeviceCdcAcmSend(s_cdcVcom.cdcAcmHandle, USB_CDC_VCOM_BULK_IN_ENDPOINT, buf, len);
}
```

- 要从USB CDC接收数据，

```
usb_status_t USB_DeviceCdcVcomCallback(class_handle_t handle, uint32_t event, void *param)
{
   switch (event)
   {
   …
           case kUSB_DeviceCdcEventRecvResponse:
           {
              if ((1 == s_cdcVcom.attach) && (1 == s_cdcVcom.startTransactions))
              {
                 uint8_t rx_size;
                 rx_size = epCbParam->length;
                 {
                     error = USB_DeviceCdcAcmRecv(handle, USB_CDC_VCOM_BULK_OUT_ENDPOINT,
cdc_rx_buf, g_UsbDeviceCdcVcomDicEndpoints[1].maxPacketSize);
                  }

                  cdc_rx_cb(cdc_rx_buf, rx_size);
               }
            }
            break;
   }
}
```

3 上手使用USBtinViewer

为了验证USB-CAN适配器的功能，在本节中，我们使用了开源软件USBtinViewer和一个商用的USB CAN适配器（PCAN-USB）。

USBtinViewer可以从https://www.fischl.de/usbtin/#usbtinviewer下载。

我们使用PCAN-USB作为商用的USB-CAN适配器，使用busmaster作为软件。

Busmaster可以从https://rbei-etas.github.io/busmaster/下载。

图2所示为硬件测试环境。

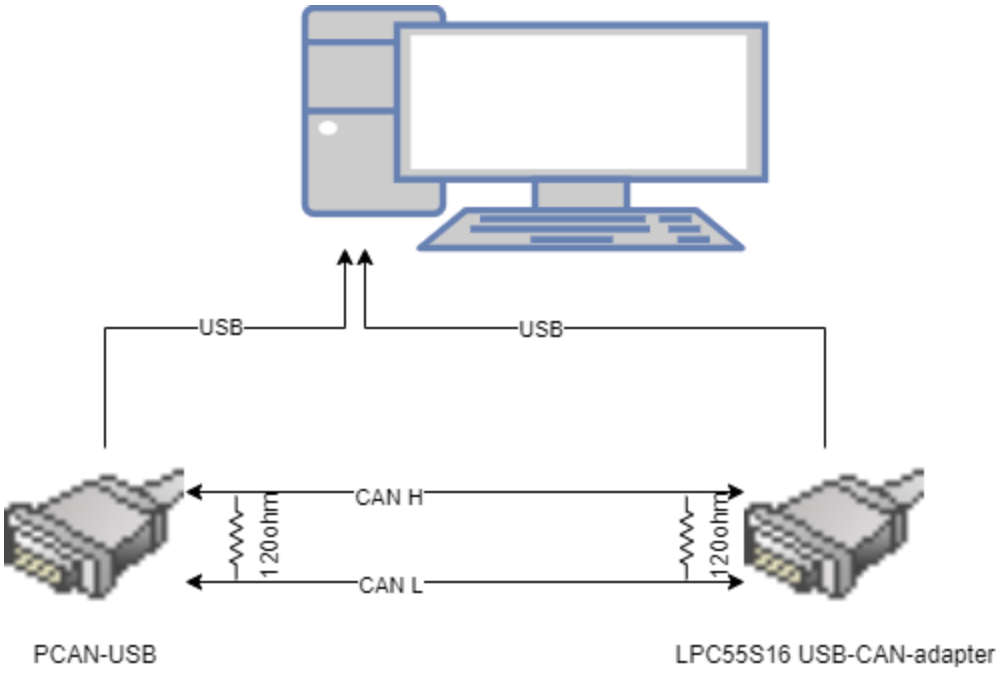

图2. 测试环境的设置

3.1 将硬件连接到USBtinViewer

1. 下载USBtinViewer，将USB-CAN适配器的USB端口连接到PC。会弹出一个USB CDC COM端口。

注意

不同PC的COM端口号不同。

图3. USB CDC端口的枚举

打开USBtinViewer，选择COM端口和CAN波特率（本例中为500 K）。点击**“连接” （Connect）**，USBtinViewer会返回固件信息，如图4所示。此信息表示连接成功。

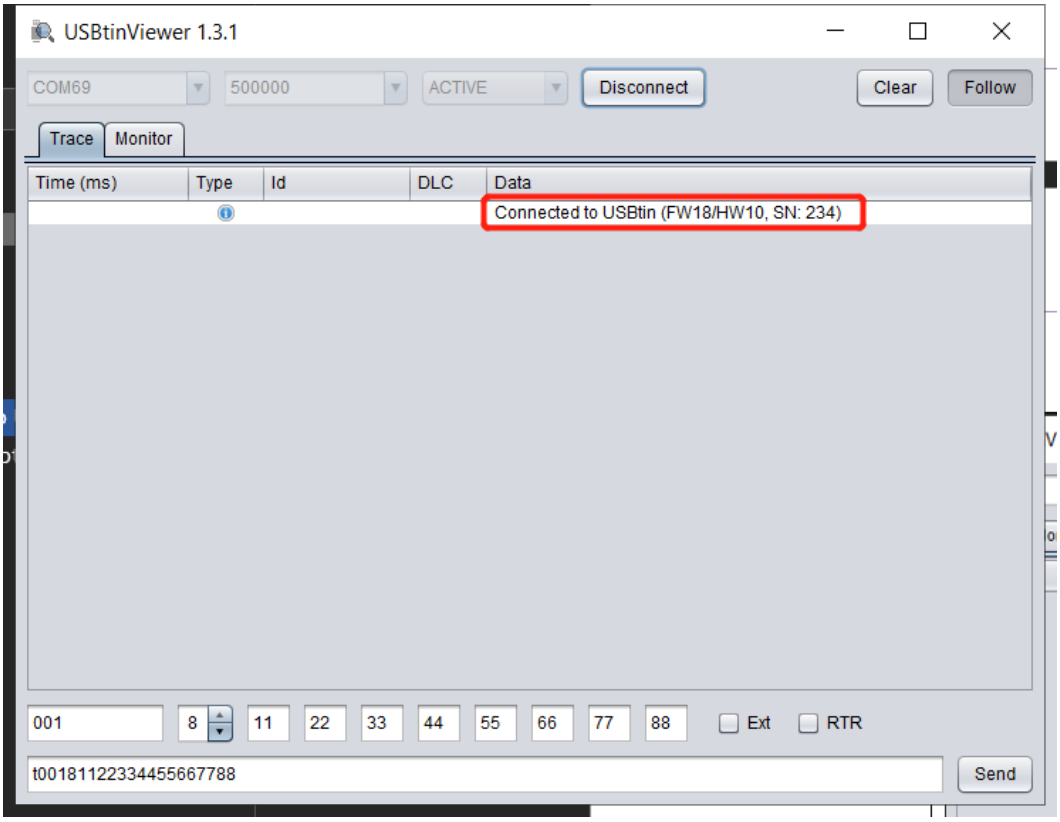

图4. 将USB-CAN适配器连接到USBtinViewer

2. 打开busmaster，连接PCAN-USB。选择500 K波特率，如图5所示。

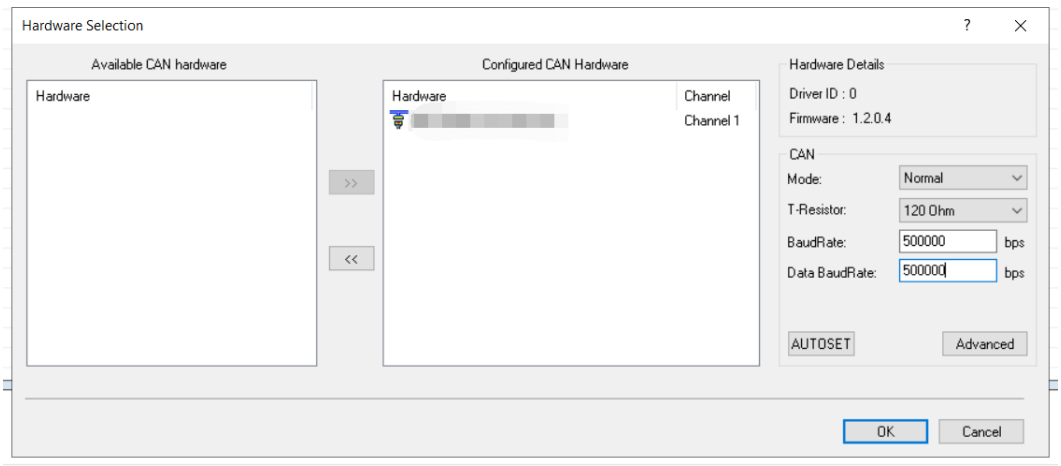

图5. 将PCAN-USB连接到busmaster

3. 从USBtinViewer发送CAN数据，并由busmaster接收。

   连接USB-CAN适配器和PCAN-USB。在USBtinViewer底部的CAN TX框中，输入CAN消息的ID、DLC和数据字段。点击**“发送” （Send）**，USB-CAN适配器将发送CAN消息。

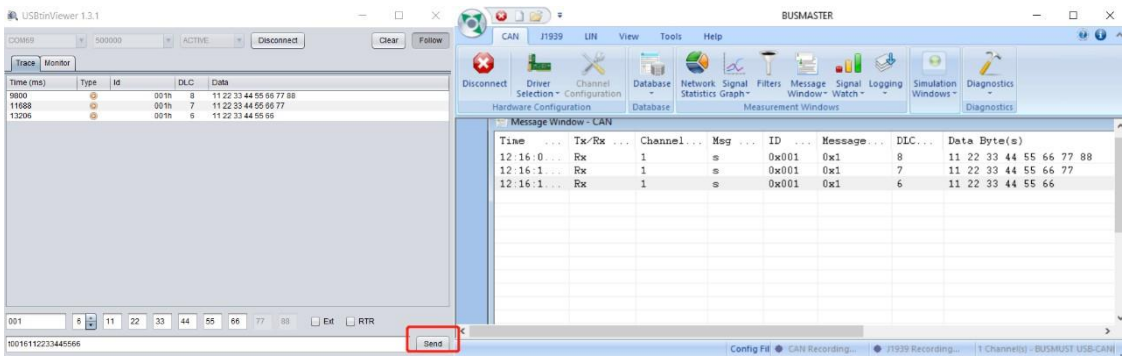

图6. 从USBtinViewer发送CAN数据，并由busmaster接收

在busmaster的**“消息”**窗口中，可以接收到相同的CAN消息，如图6所示。

4. 从busmaster发送CAN数据，并由USBtinViewer进行验证。

   在busmaster中，打开**“发送”窗口**。点击**“消息名称”（Message Name）**列下的空白区域。输入新消息的名称、DLC和帧数据字段。点击**“发送消息”（Send message）**，busmaster将发送CAN消息。在USBtinViewer上，可以监控到该CAN消息，如图8所示。

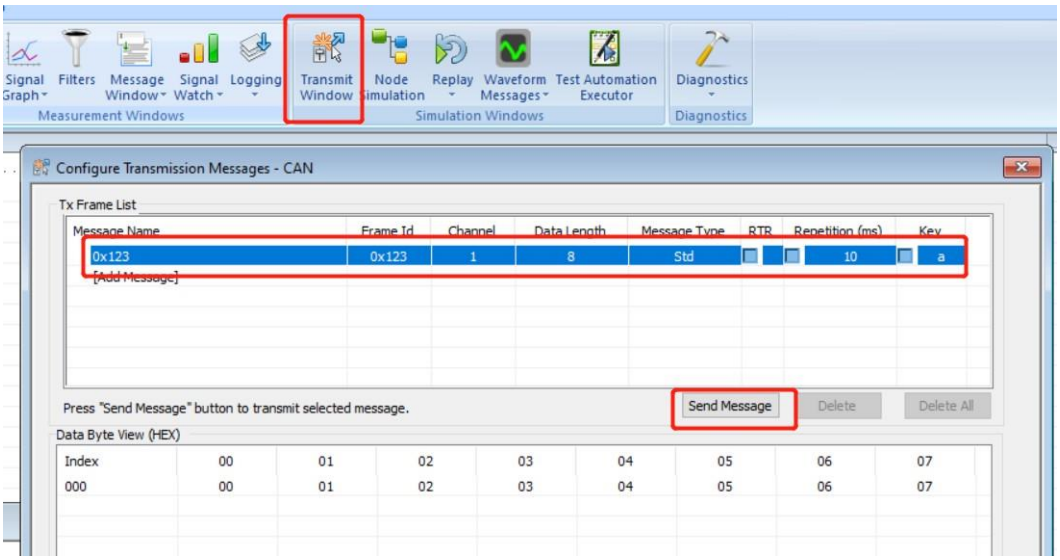

图7. 从busmaster发送CAN数据，并由USBtinViewer接收

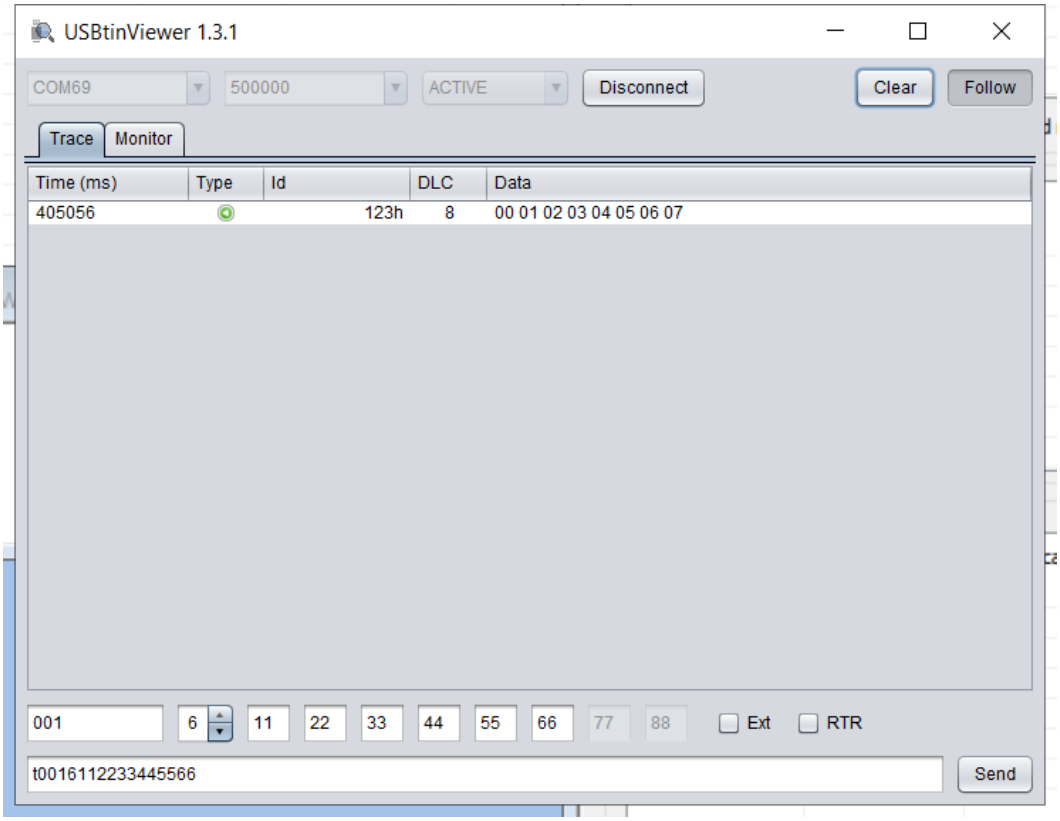

图8. USBtinViewer接收到CAN的消息

4 第三方和社区资源

USBtin网页上提供了许多实用的第三方资源。其中包括支持USBtin的库和工具。它提供了丰富的资源，并支持多种编程语言。

Third party and community projects - Software supporting USBtin

Libraries and tools supporting USBtin:

- pyUSBtin - Python implementation of the USBtin API
- CsharpUSBtinLib - C# USBTin library
- CANFlasherUTNL - Bootloader GUI for NXP LPC11C22/24 devices
- CANToolz - Framework for analysing CAN networks and devices
- CANopen API for Windows (C++, C#/.NET) by Datalink Engineering
- UsbTinQt - Qt application written in C++
- CANcool - Open Source CAN bus Analyser and Simulation Software (Pascal / Delphi 7)

图9. 第三方工具支持USBtin

5 USB-CAN适配器的原理图

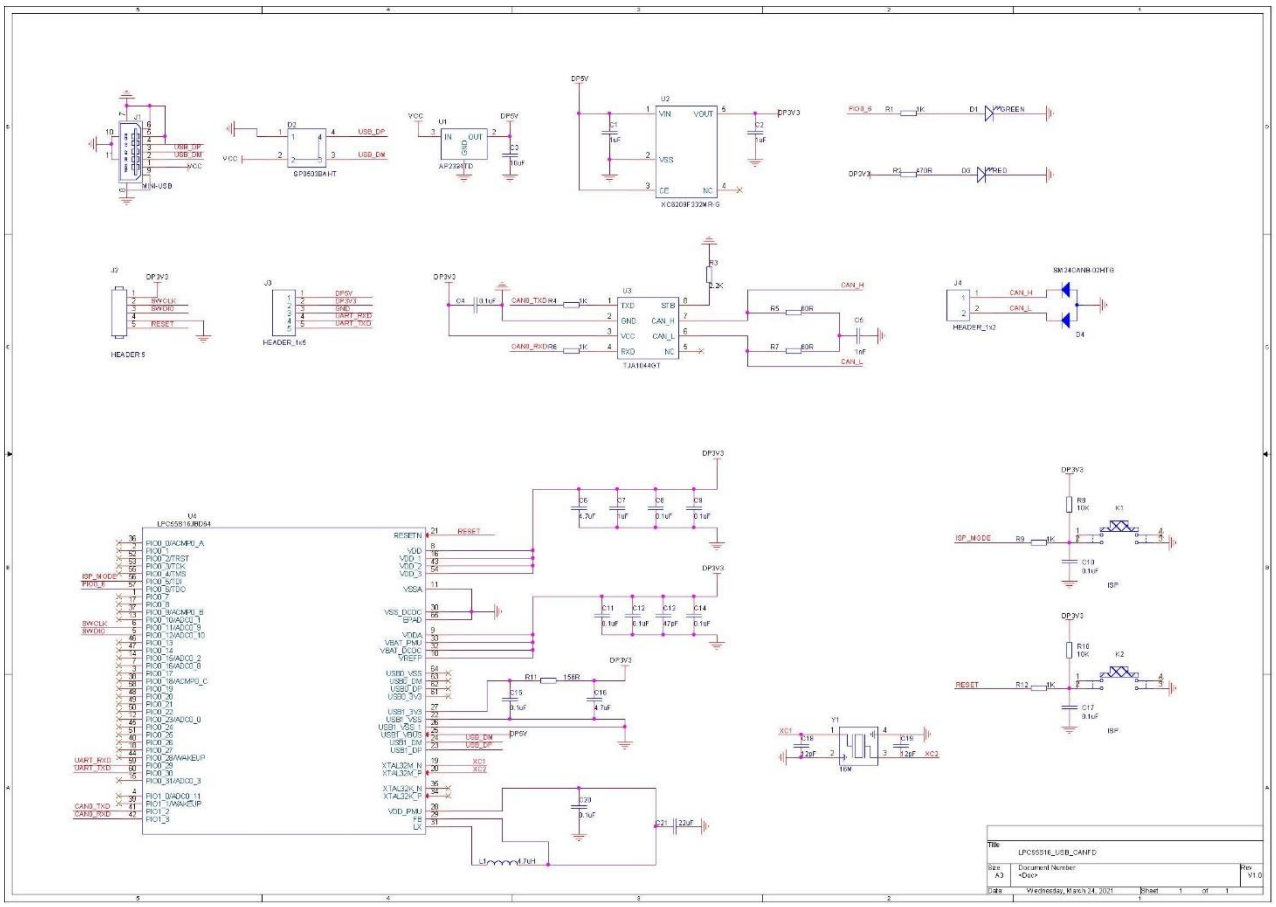

图10. USB-CAN适配器的原理图

6 参考资料

1. https://www.fischl.de/usbtin/#usbtinviewer
2. https://rbei-etas.github.io/busmaster/

7 修订历史

版本号	日期	说明
第0版	2022年1月18日	初版发布

How To Reach Us

Home Page:

nxp.com.cn

Web Support:

nxp.com.cn/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.cn/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.cn

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

Date of release: 18 January 2022

Document identifier: AN13515

arm