AN14169

基于 **i.MX RT** 系列芯片构建一个自定义类 **USB** 设备

Rev. 1 — **2024**年**3**月**8**日 应用笔记

文档信息

信息	内容
关键词	AN14169, USB, 自定义类
摘要	本应用笔记介绍了如何构建一个自定义类的 USB 设备。本应用笔记以 cdc_vcom 例程为基础，详细描述如何修改此例程以实现一个仅包含两个双向 Bulk 端点的自定义类设备，并且会描述如何从电脑端构建相关的上位机与自定义类设备进行双向通信传输数据。

1 简介

本应用笔记介绍了如何构建一个自定义类的 USB 设备。

本应用笔记以 usb_device_cdc_vcom 例程为基础，详细描述如何修改此例程以实现一个仅包含两个双向 Bulk 端点的自定义类设备，并且会描述如何从电脑端构建相关的上位机与自定义类设备进行双向通信传输数据。

2 USB 自定义类设备简介

USB 协会根据所实现的各种功能定义了很多的类，USB 设备在开发阶段都会根据功能选择使用相关的类进行开发，不管是 Windows 还是 Linux 基础的系统都已经根据 USB 协会的定义实现了大部分常规类的驱动。所以，对大多数的 USB 应用来说，照着定义开发就能实现基本的功能，并且大部分都能直接在电脑上使用。

现有的类的定义囊括了绝大多数的 USB 应用，但是在嵌入式开发中，USB 作为一种速度较高且结构简单的接口，经常会有现有类以外的数据传输需求。USB 协会在定义各个类的协议的同时也定义了一种自定义类的选项。开发者可以在接口描述符中选择 0xFF 作为类字段的值，然后再根据自己的需求定义接口描述符中的各个字段。

本篇应用笔记以一个较为简单的仅包含两个 Bulk 双向传输端点的自定义类为例，介绍自定义类以及相关上位机的实现。用户根据实际的应用需求，可以添加多个包含不同类型端点的 Interface，来实现更为复杂的数据传输需求。

3 实现自定义类设备

本篇应用笔记使用 2.14.0 版本的 RT1170-EVKB SDK 包内的 usb_device_cdc_vcom 例程作为示例，此例程中实现了两个接口，分别是用于传输控制信息的 CDC Control 接口以及用于数据传输的 CDC Data 接口。在仅仅需要双向数据通信的场景下，CDC Control 的接口可以完全移除，所以最终我们实现的自定义类设备完整 Configure 描述符仅包含四个描述符， 分别是一个设备 Configure 描述符，一个 Interface 描述符，以及两个 Endpoint 描述符。

如要自定义设备，请按以下步骤操作：

1. 改造描述符部分。
   例程原本的代码中，定义了一个宏 USB_DEVICE_CONFIG_CDC_CIC_EP_DISABLE，打开此宏定义后，CDC Control Interface 中的端点会被失能。使能此宏后，修改 g_UsbDeviceConfigurationDescriptor 此数组中的元素，将 length 项修改为如 图 1 所示。

图 1. 修改描述符长度

2. 在数组的后半部分，将 Control Interface 相关的六个描述符全部删去，只保留一共四个描述符。然后将 CDC Data Interface 描述符的 Class 部分按图 2 所示改成自定义类。

图 2. 修改 Interface 描述符

3. Interface 描述符部分修改后，还需要把用于初始化类驱动的 g_UsbDeviceCdcVcomInterfaces 数组的第一组元素删除，并且修改剩余一个 Interface 的 class 相关内容。

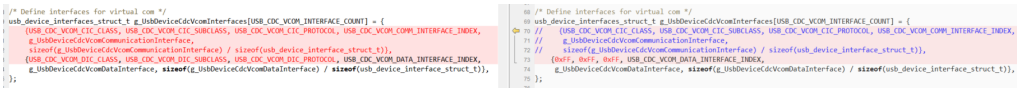

图 3. 修改 Interface 数组

4. 再把 usb_device_descriptor.h 中的宏 USB_CDC_VCOM_INTERFACE_COUNT 的值改成1，再将 USB_DEVICE_CLASS 宏的值改为 **0x00**。
完成描述符部分的修改后，需要对 CDC ACM 的类驱动进行一些修改。CDC ACM 的类驱动中的 USB_DeviceCdcAcmEndpointsInit() 接口中，首先会获取 Control 接口的相关信息，由于前面描述符中已经删除了 Control 接口，所以此处获取 Control 接口信息的部分需要删除。删除部分的代码为 usb_device_cdc_acm.c 的 217 行到 266 行。
同时在初始化数据端点时，会有一部分判断 Interface 中的类的代码，将此段代码按照 图 4 所示进行修改。

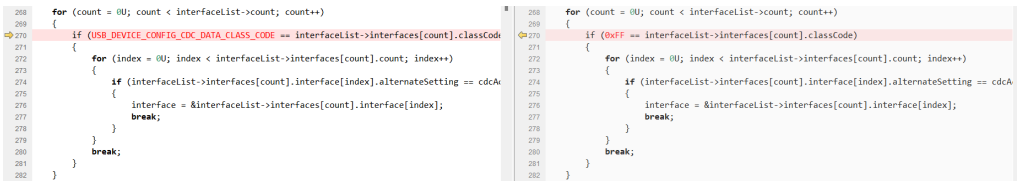

图 4. 修改类驱动

由于本应用笔记中实现的 Bulk 传输与 CDC ACM 的 data 传输部分基本一致，所以可以重用 CDC ACM 的部分类驱动。对于更加复杂的应用，用户需要重构自己的类驱动，本质上类驱动部分是为了更好的实现类的功能，它的下一层调用是通用 Controller Driver，参考现有的各种类驱动基本上可以很方便的实现自定义类的驱动。

5. 驱动修改完成后，需要对应用层也进行一些修改。应用层面上，demo 原本的设置里建立了一个开始传输的标志位 startTransactions，而此标志位只有在 CDC ACM 的相关 request 全部获取完成后才会置位，所以在 virtual_com.c 文件中，将所有如 图 5 的判断中的后半部分去除以直接实现接收到数据后回写的功能。

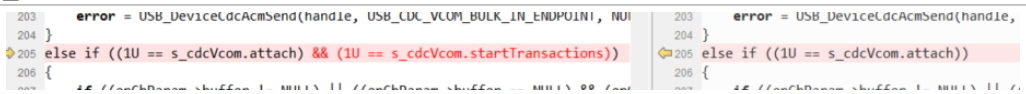

图 5. 修改应用代码

完成上述所有修改后即实现了一个仅包含两个 Bulk 数据端点的自定义类设备，在编译下载完成后直接接入电脑，会在设备管理器内显示为其他设备。

4 实现上位机

由于自定义类设备在电脑上没有现成的 Driver，所以在插入电脑后显示为其他设备，这个时候就需要编写一个上位机以与此设备进行交互。本篇应用笔记介绍的上位机使用 libusb 这个跨平台库来操作 USB 设备。

首先需要到 https://github.com/libusb/libusb/releases 此页面下载相关的库文件。在此链接内应选择 binaries 版本的压缩包，包内包含各种编译器所编译出的库文件以及 libusb 的头文件。

在使用 libusb 操作 USB 设备前，需要为相关设备更新驱动为通用 USB 驱动程序。

使用 Zadig 软件可以很方便的将设备的驱动程序更新为支持 `libusb` 库的通用驱动。从 https://github.com/pbatard/libwdi/releases 可以下载最新的 Zadig 软件。

下载完成 Zadig 软件后打开，软件中会显示出如 图 6 所示的连接的自定义类设备，此时此设备的 Driver 显示为 **None**。

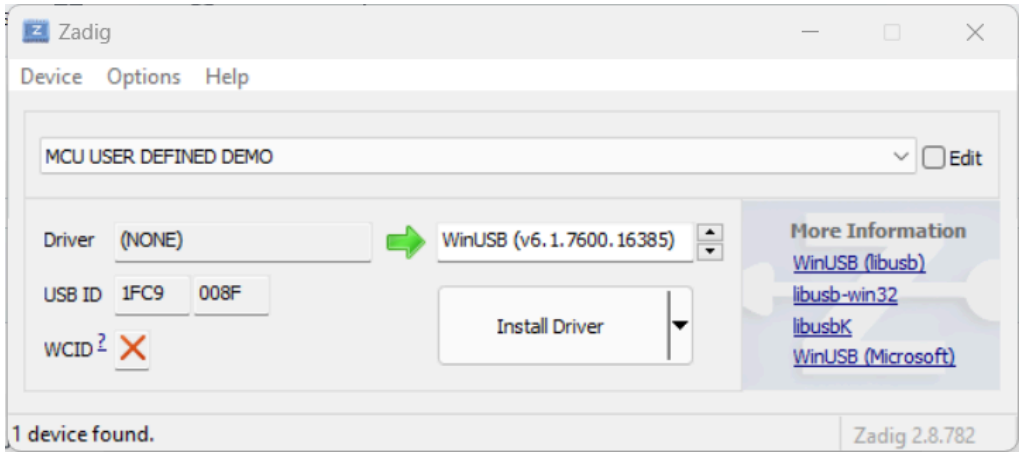

图 **6. Zadig** 软件图

在 Driver 选项中选择 WinUSB，然后点击 Install Driver 按钮，等待 Driver 安装。Driver 安装完成后，在设备管理器内的 Universal Serial Bus Device 下可以看到此自定义类设备。

设备的 Driver 安装完成后，即可开始进行上位机的开发，本应用笔记的上位机开发是基于 Visual Studio 完成的，首先安装 Visual Studio 2022 软件。并且在软件的安装界面勾选 MFC 相关组件。

安装完成后，如要创建一个空的项目，请按以下步骤操作：

1. 打开软件。
2. 新建 MFC 应用项目。
3. 在项目创建向导内的应用程序类型项上选择基于对话框。
4. 点击完成以创建空白项目。

项目创建完成后进入如 图 7 所示的控件编辑界面。

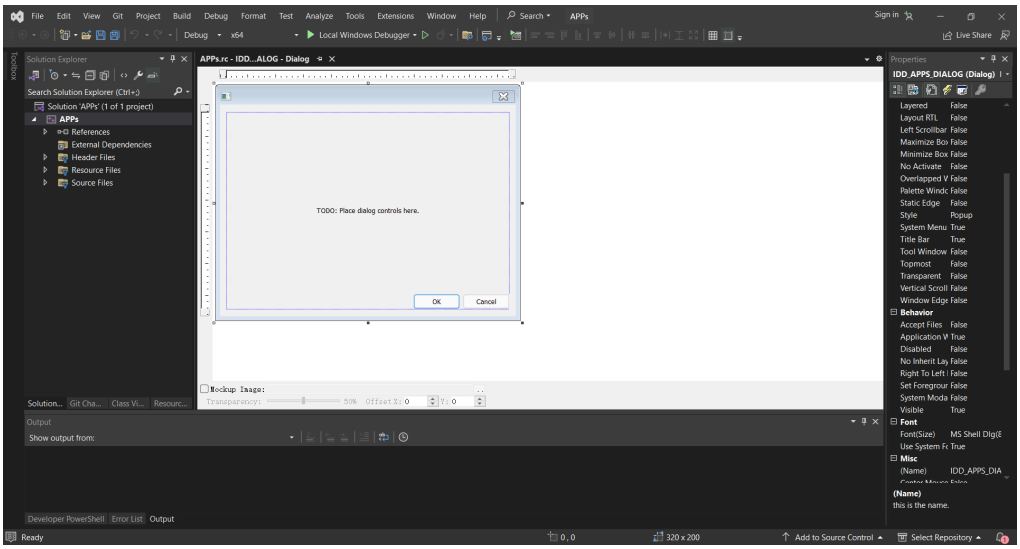

图 **7.** **MFC** 项目图

如要创建一个控件，请按以下步骤操作：

1. 在此视图下找到工具箱侧边栏，选择 Button 控件。
2. 在对话框内放置两个 Button 控件，分别作为连接按钮以及发送按钮。
3. 选择 Edit Control 控件，分别作为发送对话框以及接收显示对话框。构建出如 图 8 所示的窗体。

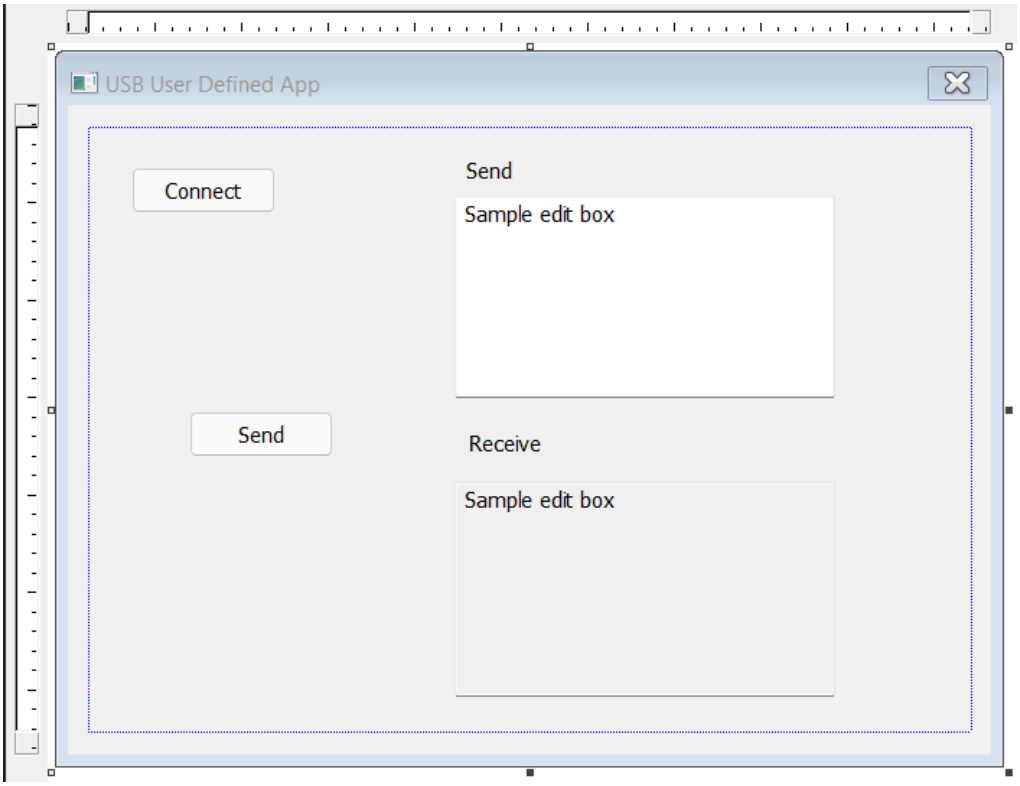

图 **8.** 控件编辑图

如要添加成员函数，请按以下步骤操作：

1. 在窗体框内鼠标右击选择类向导，如 图 9 所示。
2. 在跳出窗口内选择刚刚创建的 Connect 以及 Send 按钮，选择添加 BN_CLICKED 处理程序。

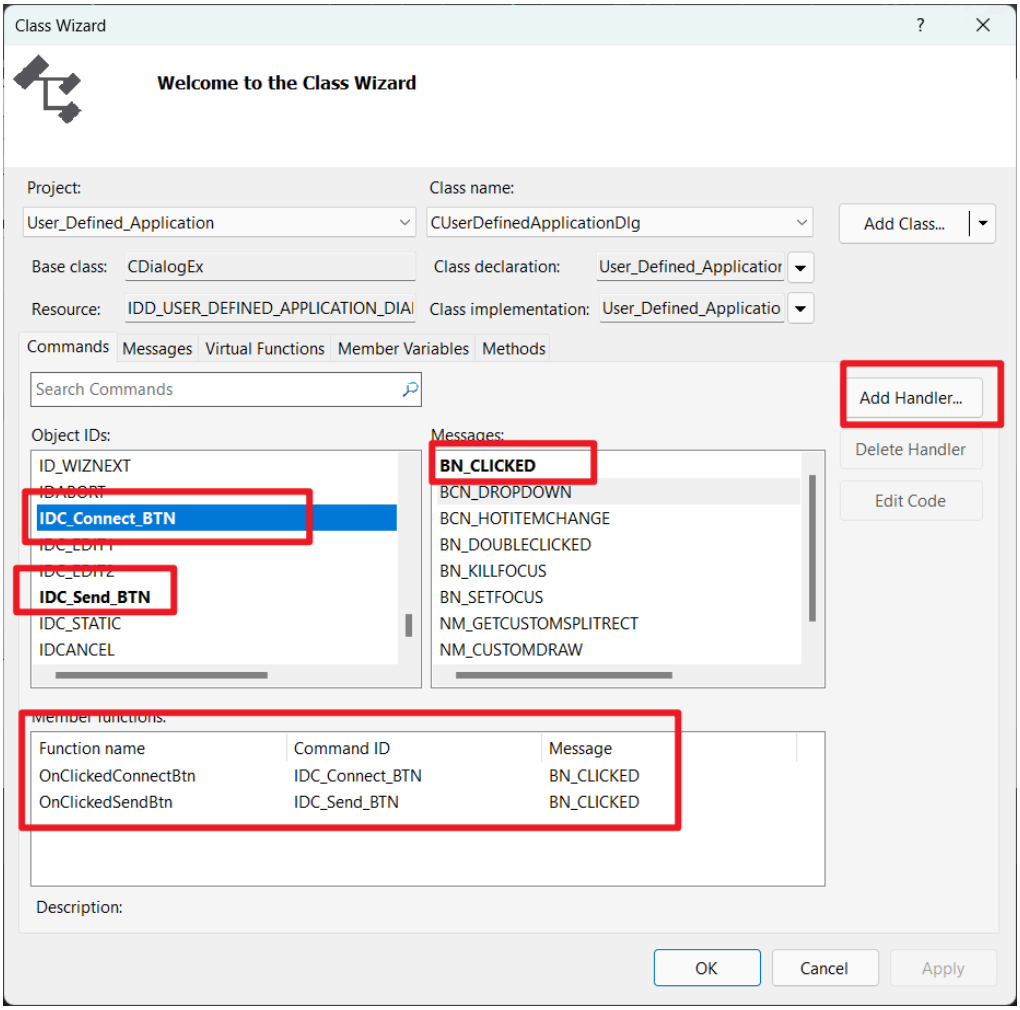

图 9. 成员函数添加图

函数添加完成后切换到解决方案资源管理器视图，在头文件中添加前面下载的 libusb 的包里的 libusb.h 文件。在项目的窗体源码文件内添加 libusb.h 头文件包含。拷贝 libusb-1.0.26-binaries\VS2015-x64\dll 目录下的 libusb-1.0.dll 以及 libusb-1.0.lib 文件到 VS 解决方案目录内。

在解决方案的属性中选择配置属性条目下的 VC++ 目录项，将 libusb-1.0.lib 所在文件夹加入库目录条目。接着在连接器条目下的输入项，在附加依赖项选项内输入 `libusb-1.0.lib`。保存设置后直接点击调试以生成 Debug 目录。此步骤中 VS 可能会报错，忽略错误后，将 libusb-1.0.dll 拷贝到解决方案目录下的 \x64\Debug\ 目录内。

完成上述步骤后开始添加代码，在 `CUserDefinedApplicationDlg` 类内添加两个成员函数，分别用于连接 USB 设备以及发送数据，创建如下的 `connect_device()` 函数。

```
libusb_device_handle* handle;
libusb_device* dev;
struct libusb_config_descriptor* conf_desc;
uint8_t endpoint_in = 0, endpoint_out = 0;  // default IN and OUT endpoints
int CUserDefinedApplicationDlg::connect_device()
{
    const struct libusb_endpoint_descriptor* endpoint;
```

```
    int i, j, k, r;
    int iface, nb_ifaces, first_iface = -1;
    struct libusb_device_descriptor dev_desc;
    libusb_init(NULL);
    handle = libusb_open_device_with_vid_pid(NULL, 0x1fc9, 0x0094);
    if (handle == NULL) {
        MessageBox(TEXT("Connect Failed!"), TEXT("message"), MB_OK);
        return -1;
    }
    dev = libusb_get_device(handle);
    CALL_CHECK_CLOSE(libusb_get_config_descriptor(dev, 0, &conf_desc), handle);
    nb_ifaces = conf_desc->bNumInterfaces;
    if (nb_ifaces > 0)
        first_iface = conf_desc->interface[0].altsetting[0].bInterfaceNumber;
    for (i = 0; i < nb_ifaces; i++) {
        for (j = 0; j < conf_desc->interface[i].num_altsetting; j++)
        {
            for (k = 0; k < conf_desc->interface[i].altsetting[j].bNumEndpoints;
 k++) {
                struct libusb_ss_endpoint_companion_descriptor* ep_comp = NULL;
                endpoint = &conf_desc->interface[i].altsetting[j].endpoint[k];
                if ((endpoint->bmAttributes & LIBUSB_TRANSFER_TYPE_MASK) &
 (LIBUSB_TRANSFER_TYPE_BULK | LIBUSB_TRANSFER_TYPE_INTERRUPT)) {
                    if (endpoint->bEndpointAddress & LIBUSB_ENDPOINT_IN) {
                        if (!endpoint_in)
                            endpoint_in = endpoint->bEndpointAddress;
                    }
                    else {
                        if (!endpoint_out)
                            endpoint_out = endpoint->bEndpointAddress;
                    }
                }
                if (ep_comp) {
                    libusb_free_ss_endpoint_companion_descriptor(ep_comp);
                }
            }
        }
    }
    libusb_free_config_descriptor(conf_desc);
    r = libusb_set_auto_detach_kernel_driver(handle, 1);
    for (iface = 0; iface < nb_ifaces; iface++)
    {
        r = libusb_claim_interface(handle, iface);
        if (r != LIBUSB_SUCCESS) {
            MessageBox(TEXT("Connect Failed!"), TEXT("message"), MB_OK);
            return -1;
        }
    }
    MessageBox(TEXT("Connect Successfully!"), TEXT("message"), MB_OK);
    return 0;
}
```

再创建 `send_data()` 函数用于发送、接收以及显示数据。

```
int CUserDefinedApplicationDlg::send_data()
{
    int r = 0;
    int size = 0;
    CString send_content;
    CString receive_content;
```

```
    CString trans;
    int len = 0;
    unsigned char data_trans[512];
    unsigned char data_recv[512];
    Send_Box.GetWindowTextW(send_content);
    len = send_content.GetLength();
    for (int j = 0; j < len; j++)
    {
        data_trans[j] = (unsigned char)send_content[j];
    }
    r = libusb_bulk_transfer(handle, endpoint_out, data_trans, len, &size, 0);
    if (r != LIBUSB_SUCCESS) {
        printf("   Failed\n");
        return -1;
    }
    int send_size = libusb_bulk_transfer(handle, endpoint_in, data_recv,
 sizeof(data_recv), &size, 2000);
    if (send_size < 0)
    {
        printf("   Failed\n");
        return -1;
    }
    for (int i = 0; i < size; i++)
    {
        trans.Format(_T("%c"), data_recv[i]);
        receive_content += trans;
    }
    Receive_Box.SetWindowTextW(receive_content);
    return 0;
}
```

完成上述操作后即可编译运行。

5 结果测试

分别实现了设备以及上位机后，即可进行测试。将 IAR 工程的代码下载进 RT1170-EVKB板，连接 USB OTG1 端口到电脑，打开PC上的上位机软件，点击 Connect，弹出连接成功对话框。在 Send 对话框内敲入字符，点击Send按钮，上位机会读取 Send 对话框内的内容并且发送给 Device，Device 收到数据后即会拷贝一份并发回给 Host，Host 接收到数据后会将数据显示在 Receive 对话框内。测试结果如 图 10 所示。

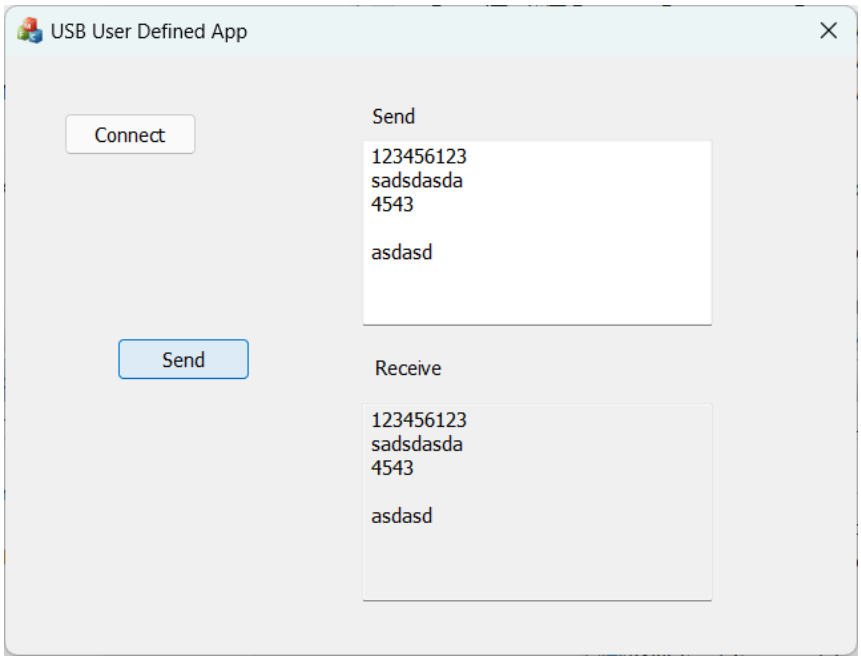

图 **10.** 测试结果图

本篇应用笔记中实现的是较为简单的 Bulk 传输的自定义类设备，并且上位机端也仅仅是发送数据接收数据后显示到对话框。对于一些需要传输数据并且保存为文件的应用，可以在上位机代码内添加相关的逻辑，将接收到的数据保存为文件。对于一些通信协议方面的应用，可以再添加包含 Interrupt 端点的 Interface，实现自定义协议的交互。

6 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 修订记录

表 1 汇总了自初始版以来对本文档所做的更改。

表 1. 修订记录

文档号	日期	说明
AN14169_ZH v.1	2024 年 3 月 8 日	初次发布

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

IAR — is a trademark of IAR Systems AB.

i.MX — is a trademark of NXP B.V.

基于 i.MX RT 系列芯片构建一个自定义类 USB 设备

内容

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

Date of release: 2024年3月8日
Document identifier: AN14169_ZH