

## AT32 USB Multi Bridge

## Introduction

AT32 USB multi-function bridge provides a dedicated high-speed USB interface to communicate with the target through interfaces such as UART, CAN, SPI, I<sup>2</sup>C and RS485. The USB transfers peripheral data according to the CDC protocol, which can be implemented in conjunction with USART IAP, CAN IAP, SPI IAP, I<sup>2</sup>C IAP. The USB CDC driver needs to be installed in some systems.

Applicable peripherals:

- USB to USART
- USB to CAN
- USB to SPI
- USB to I<sup>2</sup>C
- USB to RS485

Applicable products:

Part number	AT32F405 series
-------------	-----------------

## Contents

<b>1</b>	<b>Overview .....</b>	<b>5</b>
1.1	USB bridge features .....	5
1.1.1	Select a peripheral .....	5
1.1.2	Modify baud rate.....	7
1.1.3	Hardware .....	8
1.1.4	Software .....	9
1.2	Demo project.....	9
<b>2</b>	<b>USB-to-USART .....</b>	<b>10</b>
2.1	Connection .....	10
<b>3</b>	<b>USB-to-CAN.....</b>	<b>11</b>
3.1	Connection .....	11
3.2	Data transfer between USB and CAN .....	11
<b>4</b>	<b>USB-to-SPI.....</b>	<b>13</b>
4.1	Connection .....	13
4.2	Data transfer between USB and SPI .....	13
<b>5</b>	<b>USB-to-I<sup>2</sup>C.....</b>	<b>14</b>
5.1	Connnection .....	错误!未定义书签。
5.2	Data transfer between USB and I <sup>2</sup> C interface.....	14
<b>6</b>	<b>USB-to-RS485.....</b>	<b>15</b>
6.1	Connection .....	15
<b>7</b>	<b>Revision history.....</b>	<b>16</b>

## List of tables

Table 1. Baud rate for CAN .....	7
Table 2. Clock frequency for SPI .....	8
Table 3. Clock frequency for I <sup>2</sup> C .....	8
Table 4. Document revision history.....	16

## List of figures

Figure 1. USB SETUP request format.....	6
Figure 2. SET LINE CODING command .....	7
Figure 3. Line Coding Structure.....	7
Figure 4. AT32-USBHS-Adaptor .....	8
Figure 5. Connection diagram .....	9
Figure 6. Connection between USB adaptor and USART .....	10
Figure 7. Connection between USB adaptor and CAN .....	11
Figure 8. Connection between USB adaptor and SPI .....	13
Figure 9. Connection between USB adaptor and I <sup>2</sup> C.....	14
Figure 10. Connection between USB adaptor and RS485 interface.....	15

# 1 Overview

## 1.1 USB bridge features

The AT32 device emulates a COM port according to the USBHS CDC protocol to support several peripherals. User can select one peripheral to be used at a time through USB control request and change the peripheral baud rate through the “SET LINE CODEING” command.

Applicable peripherals include:

- USB to USART
- USB to CAN
- USB to SPI
- USB to I<sup>2</sup>C
- USB to RS485

### 1.1.1 Select a peripheral

Users can select a peripheral by one of the following means:

- Select the desired peripheral during code initialization and do not change it anymore.
- Select the current peripheral in a real-time manner through USB endpoint SETUP request.
- Select the current peripheral through USB HID command.

#### 1.1.1.1 Select a peripheral during initialization

During initialization, the USB Multi Bridge can set the current interface through “usb\_multi\_bridge\_set\_type(USB\_TO\_USART);” function, including:

USB\_TO\_USART

USB\_TO\_CAN

USB\_TO\_SPI

USB\_TO\_I2C

USB\_TO\_RS485

USB\_TO\_IDLE (idle state, no peripheral connected)

#### 1.1.1.2 Select a peripheral through USB control request

In addition, users can select the desired peripheral through USB endpoint 0 SETUP command. Refer to *Universal Serial Bus Specification 2.0*. The SETUP packet format is shown below.

Figure 1. USB SETUP request format

Offset	Field	Size	Value	Description
0	<i>bmRequestType</i>	1	Bitmap	Characteristics of request:  D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host  D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved  D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	<i>bRequest</i>	1	Value	Specific request (refer to Table 9-3)
2	<i>wValue</i>	2	Value	Word-sized field that varies according to request
4	<i>wIndex</i>	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	<i>wLength</i>	2	Count	Number of bytes to transfer if there is a Data stage

## USB-TO-USART enable:

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>
0x40	0x01	0x0000	0x0000	0x0000

## USB-TO-CAN enable:

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>
0x40	0x01	0x0001	0x0000	0x0000

## USB-TO-SPI enable:

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>
0x40	0x01	0x0002	0x0000	0x0000

USB-TO-I<sup>2</sup>C enable:

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>
0x40	0x01	0x0003	0x0000	0x0000

## USB-TO-RS485 enable:

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>
0x40	0x01	0x0004	0x0000	0x0000

### 1.1.1.3 Select a peripheral through HID command

Besides, users can set the peripheral through the device HID command (HID VID (0x2E3C), PID(0xFF01)).

Set the host (HID data) to select a peripheral, as shown below:

HID data	USB bridge
0xA1 0x00	USB TO USART
0xA1 0x01	USB TO CAN
0xA1 0x02	USB TO SPI
0xA1 0x03	USB TO I2C
0xA1 0x04	USB TO RS485

### 1.1.2 Modify baud rate

The baud rate or frequency varies from peripherals, which can be set through the SET LINE CODING command.

Note: Considering the external circuit configuration, it is possible to reduce the communication rate in case of communication failure.

Figure 2. SET LINE CODING command

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_LINE_CODING	Zero	Interface	Size of Structure	Line Coding Structure

Line coding structure: Transmit the baud rate through “dwDTERate”.

Figure 3. Line Coding Structure

Offset	Field	Size	Value	Description
0	<i>dwDTERate</i>	4	Number	Data terminal rate, in bits per second.
4	<i>bCharFormat</i>	1	Number	Stop bits 0 - 1 Stop bit 1 - 1.5 Stop bits 2 - 2 Stop bits
5	<i>bParityType</i>	1	Number	Parity 0 - None 1 - Odd 2 - Even 3 - Mark 4 - Space
6	<i>bDataBits</i>	1	Number	Data bits (5, 6, 7, 8 or 16).

USART and RS485: 1600 bps~6 Mbps

CAN:

Table 1. Baud rate for CAN

Baud rate	DwDTERate
1 Mbps	1000000
500 kbps	500000
250 kbps	250000
125 kbps	125000

**SPI:**

**Table 2. Clock frequency for SPI**

Clock frequency	DwDTERate
13.5 MHz	13500000
6.75 MHz	6750000
3.375 MHz	3375000
1.6875 MHz	1687500
0.84375 MHz	843750

**I<sup>2</sup>C:**

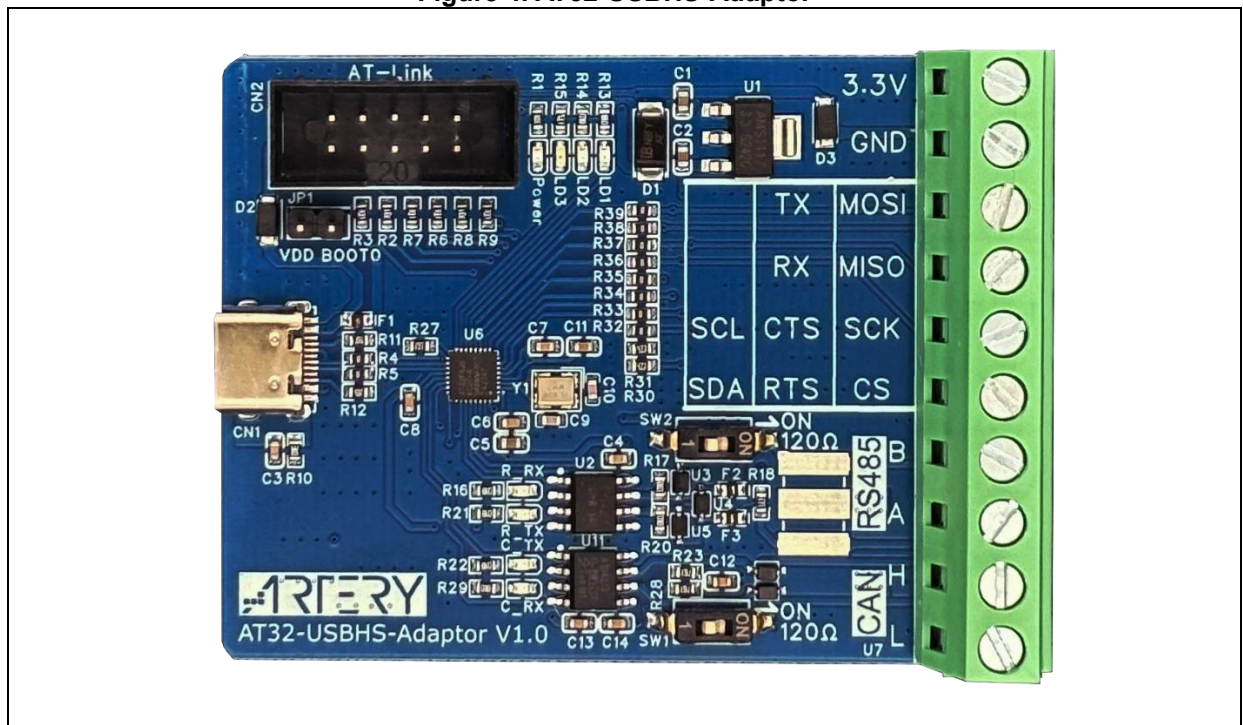
**Table 3. Clock frequency for I<sup>2</sup>C**

Clock frequency	DwDTERate
1 MHz	1000000
400 kHz	400000
200 kHz	200000
100 kHz	100000
50 kHz	50000
10 kHz	10000

## 1.1.3 Hardware

1. USB adaptor (AT32-USBHS-Adaptor)
2. USB cable

**Figure 4. AT32-USBHS-Adaptor**



3. Power supply: The device is powered by 5V through CN1 USB interface, and it supplies the external with 3.3V/350mA power.



## 4. LED

- Power: Red LED indicates that the AT32 USBHS Adaptor is powered.
- Mode indicator: LD1, LD2 and LD3, as detailed below.

MODE	LD1	LD2	LD3
USB to USART	√		
USB to SPI		√	
USB to I2C			√
USB to CAN	√	√	
USB to RS485	√		√

## 1.1.4 Software

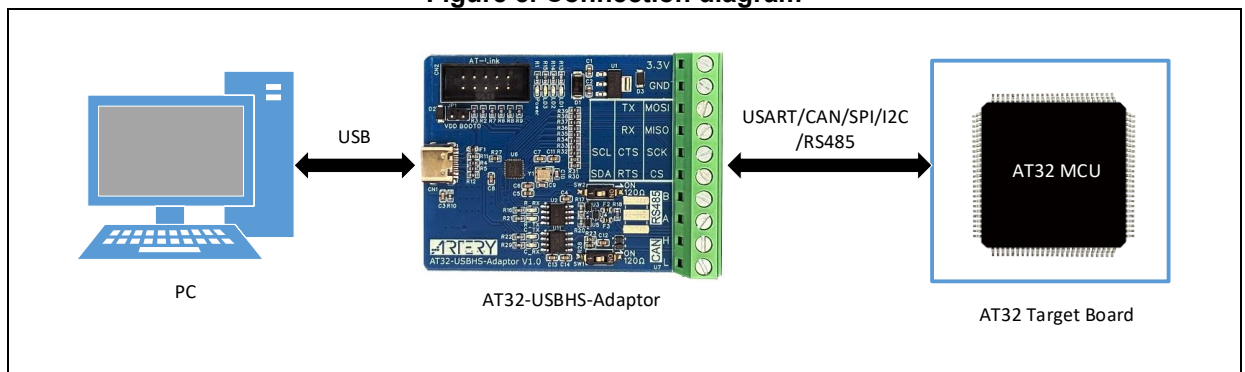
- 1) SourceCode\utilities\usb\_multi\_bridge

*Note: The demo project is built around Keil v5. If users want to use them in other compiling environments, please refer to AT32F403A\_407\_Firmware\_Library\_V2.x.x\project\at\_start\_f403a\templates (such as IAR6/7/8, Keil 4/5, eclipse\_gcc) for a simple change.*

## 1.2 Demo project

- 1) Open the “usb\_multi\_bridge” source program, and select the desired peripheral by calling “usb\_multi\_bridge\_set\_type(USB\_TO\_USART)”. Then, compile and download to the USB adaptor.
- 2) Connect USB cable to PC, and then a virtual COM device can be found in PC Manager, which can be debugged through the host virtual serial port tool.

**Figure 5. Connection diagram**



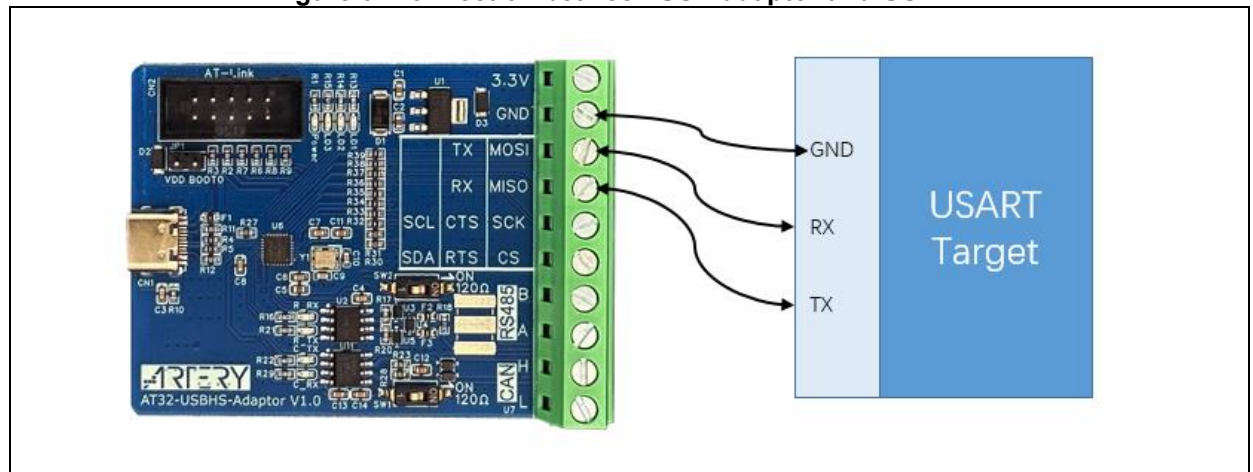
## 2 USB-to-USART

USB-to-USART bridge realizes a virtual serial port device on USB end. The USB adaptor communicates with the PC host through USB and with the upper computer through USART, thus to implement data transfer between USB and USART. Note that the adaptor USART baud rate should be the same as that of the lower computer USART.

### 2.1 Connection

The USB-to-USART bridge is realized through AT32 USBHS Adaptor. The adaptor RX and TX are connected to the target board TX and RX, respectively.

**Figure 6. Connection between USB adaptor and USART**



Relevant data formats:

Data bit	Stop bit	Parity check	Baud rate
7, 8, 9	1, 1.5, 2	Odd, even, none	1600bps~6Mbps

Refer to Section 1.1.2 for modification of baud rate and data.

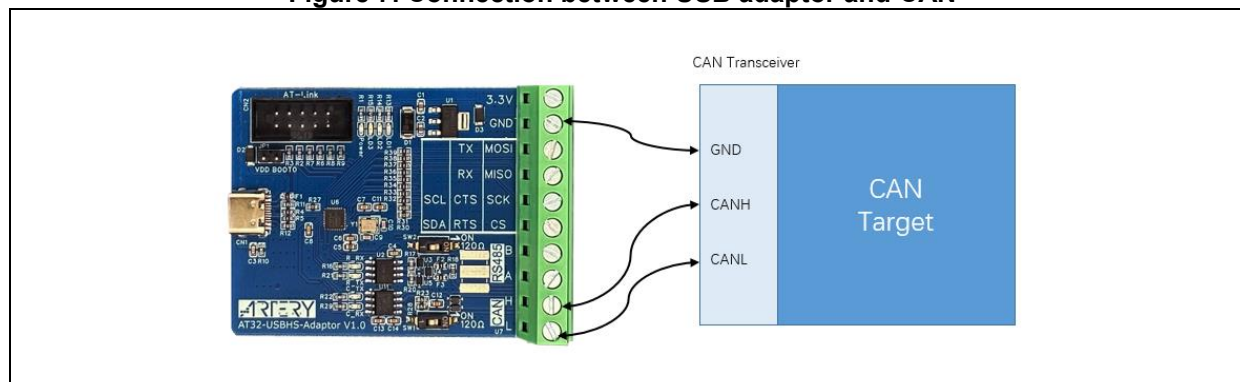
## 3 USB-to-CAN

USB-to-CAN bridge realizes a virtual serial port device on USB end. The USB adaptor communicates with the PC host through USB and with the upper computer through CAN, thus to implement data transfer between USB and CAN. Note that the adaptor CAN baud rate should be the same as that of the lower computer CAN. The CAN baud rate is 500 kbps, by default.

### 3.1 Connection

The USB-to-CAN bridge is realized through AT32 USBHS Adaptor. The adaptor CANH and CANL are connected to the target board CANH and CANL, respectively.

**Figure 7. Connection between USB adaptor and CAN**



### 3.2 Data transfer between USB and CAN

The USB-to-CAN bridge is packaged according to CAN protocol, and the format is as follows:

Frame ID (4 bytes)	ID type (1 byte)	Frame type (1 byte)	Frame length (1 byte)	Data (n byte(s), n<=8)
-----------------------	---------------------	------------------------	--------------------------	---------------------------

**Frame ID:** support 11-bit standard frame ID and 29-bit extended frame ID; LSB first

**ID type:** 0 stands for standard ID (the lower 11 bits are valid), and 1 stands for extended ID (the lower 29 bits are valid)

**Frame type:** 0 stands for data frame, and 1 stands for remote frame

**Frame length:** the frame size is less than 8 according to the CAN2.0 standard

**Data:** frame data

Refer to Section 1.1.2 for CAN baud rate.

#### Data transfer from USB to CAN bus

One USB data packet is converted into a frame of CAN data.

Received USB data packet, for example, 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x11 0x22

The first 4 bytes (0x00 0x00 0x00 0x00): frame ID (MSB)

The 5<sup>th</sup> byte (0x00): standard frame ID, lower 11 bits valid

The 6<sup>th</sup> byte (0x00): data frame

The 7<sup>th</sup> byte (0x02): data frame length

The 8<sup>th</sup> and 9<sup>th</sup> bytes (0x11,0x22): specific frame data

Convert to a frame of CAN data:

Standard frame ID=0x0001

DLC=0x02

DATA0=0x11

DATA1=0x22

**Data transfer from CAN bus to USB**

A frame of CAN data is converted to one USB data packet.

Received one standard data frame, for example, ID=0x02, DLC=0x03, DATA0=0x11, DATA0=0x22, DATA0=0x33

Convert to one USB data packet: 0x00 0x00 0x00 0x02 0x00 0x00 0x03 0x11,0x22,0x33

The first 4 bytes (0x00 0x00 0x00 0x02): standard frame ID

The 5<sup>th</sup> byte (0x00): standard frame ID

The 6<sup>th</sup> byte (0x00): data frame

The 7<sup>th</sup> byte (0x03): data frame length

The 8<sup>th</sup>, 9<sup>th</sup> and 10<sup>th</sup> bytes (0x11,0x22 0x33) specific frame data

## 4 USB-to-SPI

USB-to-SPI bridge realizes a virtual serial port device on USB end. The USB adaptor communicates with the PC host through USB and with the upper computer through SPI, thus to implement data transfer between USB and SPI. Note that the adaptor SPI serves as the master.

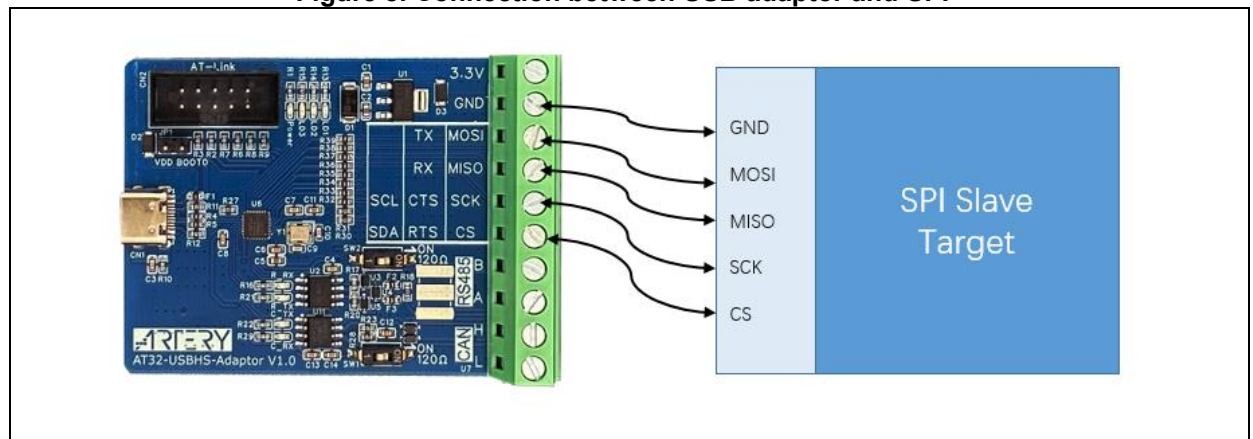
SPI master configuration:

- Set SPI master mode
- Full-duplex mode
- 8-bit MSB
- Polarity: CPOL High, CPHA low, NSS software

### 4.1 Connection

The USB-to-SPI bridge is realized through AT32 USBHS Adaptor. The adaptor MOSI, MISO, SCK and CS are connected to the target board MOSI, MISO, SCK and CS, respectively.

Figure 8. Connection between USB adaptor and SPI



### 4.2 Data transfer between USB and SPI

SPI transmits and receives data in full-duplex mode. A packet contains up to 512-byte data. If the data is greater than 512 bytes, it needs to be divided into several packets for transmission.

#### Host PC data transmit:

Example:

The host PC transmits 8 data to the target board, i.e., 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08.

1. The host PC sends 8 bytes (0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08) directly;
2. The host PC needs to read another 8 bytes of dummy data.

#### Host PC data read:

Example:

The host PC reads 8 data from the target board.

1. The host PC sends 8 bytes of dummy data;
2. The host PC reads 8 bytes of data from the target board.

Refer to Section 1.1.2 for SPI clock frequency.

## 5 USB-to-I<sup>2</sup>C

USB-to-I<sup>2</sup>C bridge realizes a virtual serial port device on USB end. The USB adaptor communicates with the PC host through USB and with the upper computer through I<sup>2</sup>C, thus to implement data transfer between USB and I<sup>2</sup>C. Note that the adaptor I<sup>2</sup>C serves as the master, and the I<sup>2</sup>C master address is 0x0C.

I<sup>2</sup>C configuration:

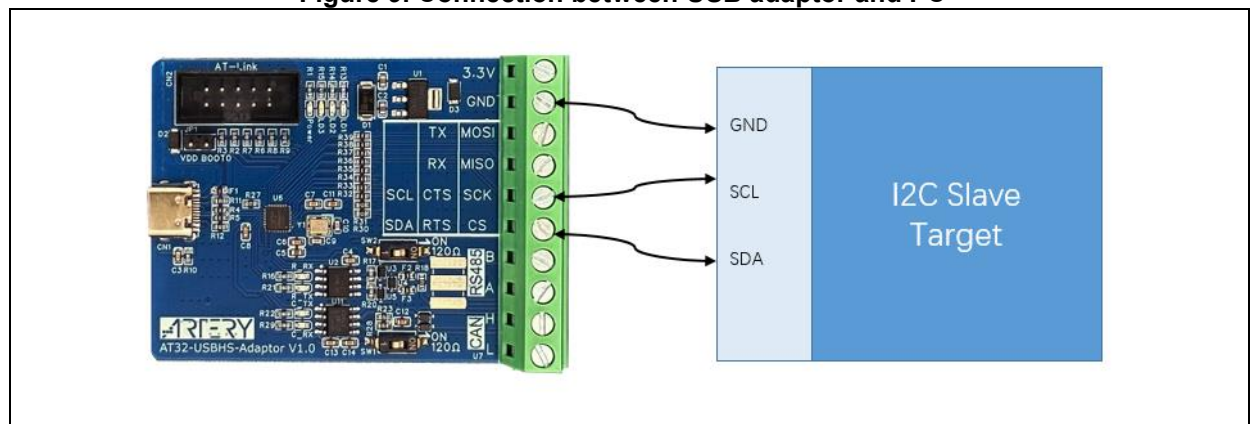
- Set I<sup>2</sup>C master mode
- Address: 0x0C

### 5.1 Connection

The USB-to-I<sup>2</sup>C bridge is realized through AT32 USBHS Adaptor. The adaptor SCL and SDA are connected to the target board SCL and SDA, respectively.

By default, the AT32 USBHS Adaptor is configured with 4.7 kΩ pull-up resistors (R30/R31) on SCL and SDA lines.

Figure 9. Connection between USB adaptor and I<sup>2</sup>C



### 5.2 Data transfer between USB and I<sup>2</sup>C interface

Use I<sup>2</sup>C interface to Transmit and receive data through I<sup>2</sup>C interface. One packet contains a maximum of 512 bytes of data. If the data is more than 512 bytes, it should be transmitted in several packets.

USB-to-I<sup>2</sup>C data package format:

Direction (1 byte)	Address (2 bytes)	Data transfer length (2 bytes)	Data (n byte)
-----------------------	----------------------	-----------------------------------	------------------

**Direction:** 0x55 stands for sending I<sup>2</sup>C data to the target board; 0xAA stands for reading I<sup>2</sup>C data from the target board.

**Address:** 2-byte target board address (MSB)

**Data length:** the length of data to be sent or read, which is less than or equal to 507 bytes (MSB)

**Data:** the data to be sent or read

Refer to Section 1.1.2 for I<sup>2</sup>C clock frequency.

**Host PC data transmit:**

Example: The host PC sends 8 data to the target board (address: 0xA0), i.e., 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

Host PC sends a packet: 0x55 0x00 0xA0 0x00 0x08 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

**Host PC data read:**

Example: The host PC reads 8 data from the target board (address: 0x2C).

1. The host PC sends 0xAA 0x00 0xA0 0x00 0x08, which means that it reads 8 data;

2. The host PC read and parse the data returned by USB, i.e., 0xAA 0x00 0xA0 0x00 0x08 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

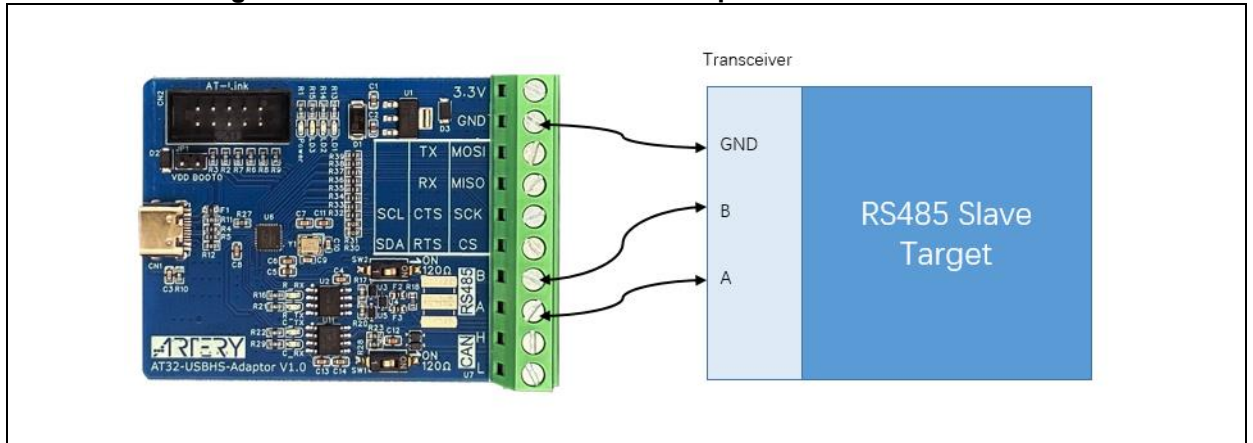
## 6 USB-to-RS485

USB-to-RS485 bridge realizes a virtual serial port device on USB end. The USB adaptor communicates with the PC host through USB and with the upper computer through RS485 interface, thus to implement data transfer between USB and RS485 interface.

### 6.1 Connection

The USB-to-RS485 bridge is realized through AT32 USBHS Adaptor. The adaptor RS485-A and RS485-B are connected to the target board A and B, respectively.

**Figure 10. Connection between USB adaptor and RS485 interface**



## 7 Revision history

Table 4. Document revision history

Date	Version	Revision note
2024.07.18	2.0.0	Initial release.
2024.08.20	2.0.1	Optimized document description. Added description of LED mode.



**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Aerospace applications or environment; (D) Weapons, and/or (E) Other applications that may cause injuries, deaths or property damages. Since ARTERY products are not intended for the above -mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form

© 2024 ARTERY Technology – All Rights Reserved