



GWU2U Driver (libusb based)

User Guide

UG1006-1.1E, 12/23/2021

Copyright © 2021 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN, Gowin, and GOWINSEMI are trademarks of Guangdong Gowin Semiconductor Corporation and are registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any denotes, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. All information in this document should be treated as preliminary. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
06/29/2021	1.0E	Initial version published.
12/23/2021	1.1E	Chapter 3 “libusb Programming Guide” updated.

Contents

Contents	i
List of Figures.....	ii
List of Tables.....	iii
1 GWU2U driver (libusb + WinUSB)	1
1.1 Use Zadig to Install Driver	1
2 Uninstall Driver	3
3 libusb Programming Guide	5
3.1 Parameter Configuration	5
3.2 SW API Introduction	6
3.3 libusb Function Library Development Flow	8
3.3.1 libusb Initialization and Exit	8
3.3.2 Open the Specified USB Device.....	8
3.3.3 Interface Declaration	11
3.3.4 Serial Communication Parameters Configuration	11
3.3.5 Synchronous Data Transmit	12
3.3.6 Synchronous Data Receive	12
3.3.7 Asynchronous Data Transfer	13
4 Programming Example	15
Terminology and Abbreviations	20
Platform Supported	21
Support and Feedback.....	22

List of Figures

Figure 1-1 List All Devices 1

Figure 1-2 Select the Device that Requires Driver Installation 2

Figure 1-3 Select the Driver Program to be Installed 2

Figure 2-1 Open Device Manager 3

Figure 2-2 Uninstall Device 4

List of Tables

Table A-1 Terminology and Abbreviations	20
Table A-2 Platform Supported	21

1 GWU2U driver (libusb + WinUSB)

GWU2U can be programmed using libusb, open source USB function library.

If you program with this function library on Windows, you need to install the WINDOWS USB driver, WinUSB.sys. It can realize data receive/transmit and the setting of parameters such as baud rate, parity bit, stop bit and byte bit width.

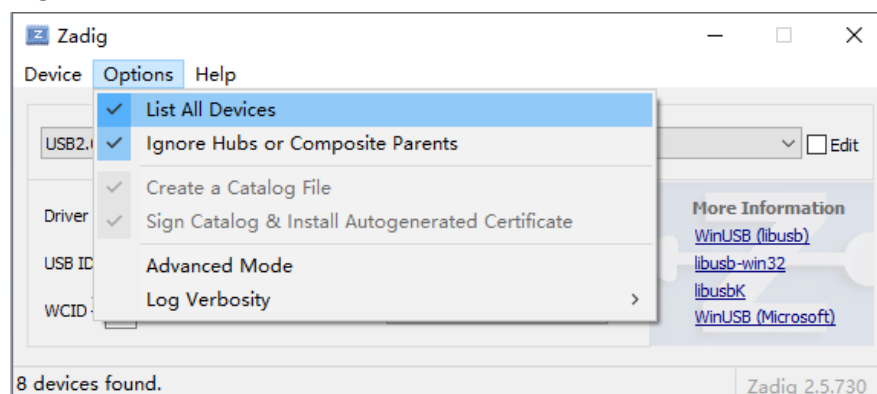
Use Gowin driver installation tool or Zadig (<https://zadig.akeo.ie/>), open source driver installation tool, to install driver. The driver installation requires administrator privileges.

If you program with this function library on Linux, you do not need to install the driver. Please skip Chapter 1 and Chapter 2.

1.1 Use Zadig to Install Driver

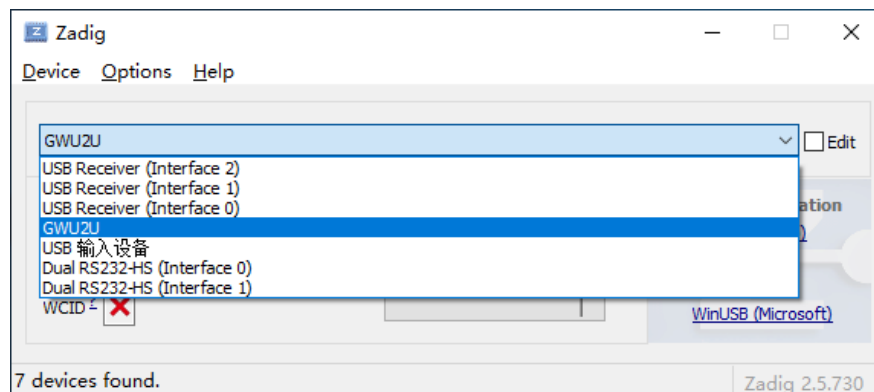
Connect GWU2U device to computer with USB interface, double-click to open Zadig (administrator privileges required), click Options, check the "List All Device" option, and all USB devices connected to the computer will be listed.

Figure 1-1 List All Devices



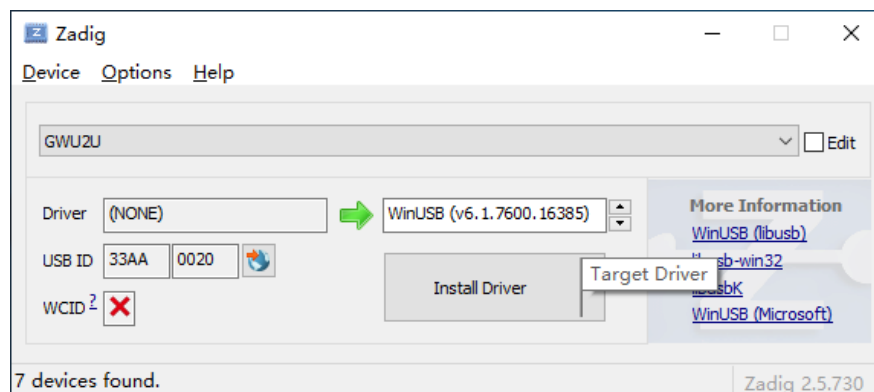
Select Gowin U2U, the device that requires driver installation.

Figure 1-2 Select the Device that Requires Driver Installation



Select the driver to be installed, use libusb+WinUSB, and select WinUSB.

Figure 1-3 Select the Driver Program to be Installed



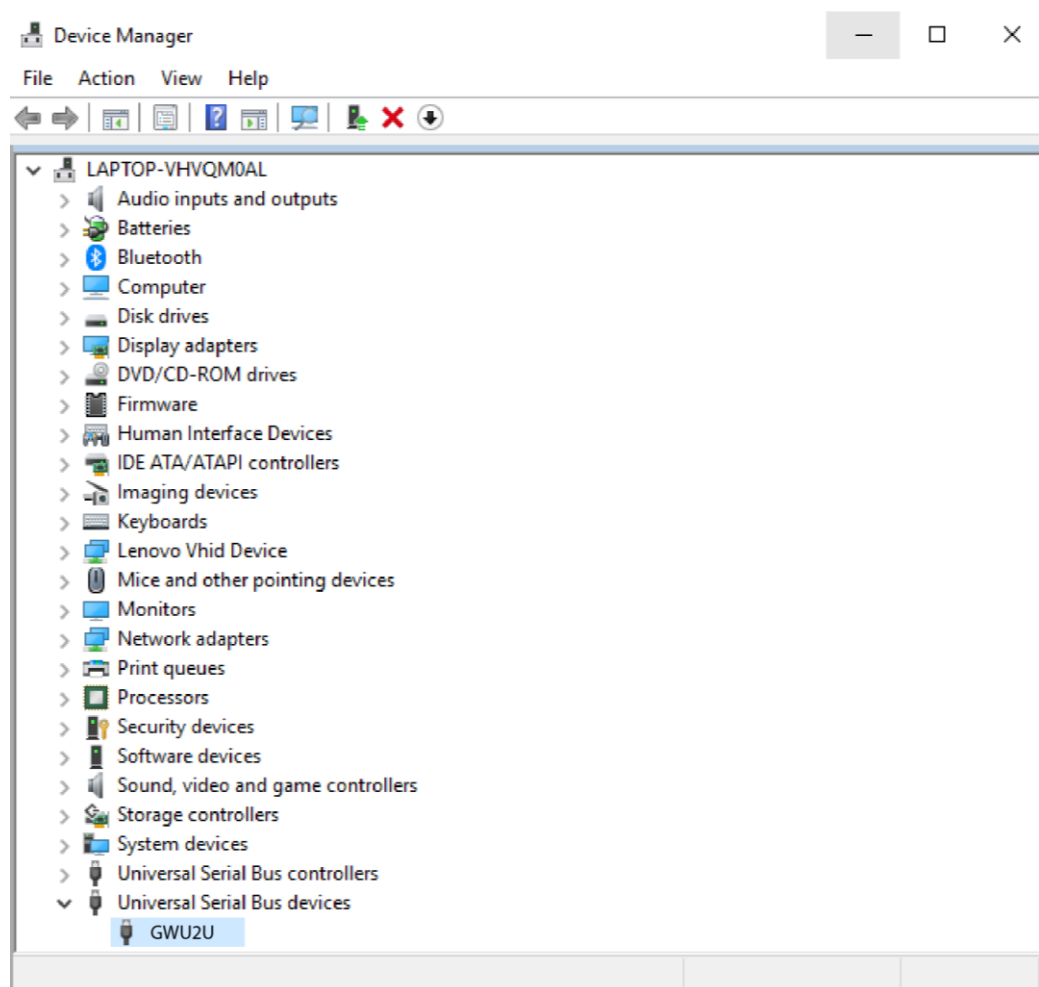
Click "Install Driver". The driver will be installed after a few moments.

Note that the button displays "Install Driver" if the driver is not currently installed, and "Replace Driver" if another driver is currently installed.

2 Uninstall Driver

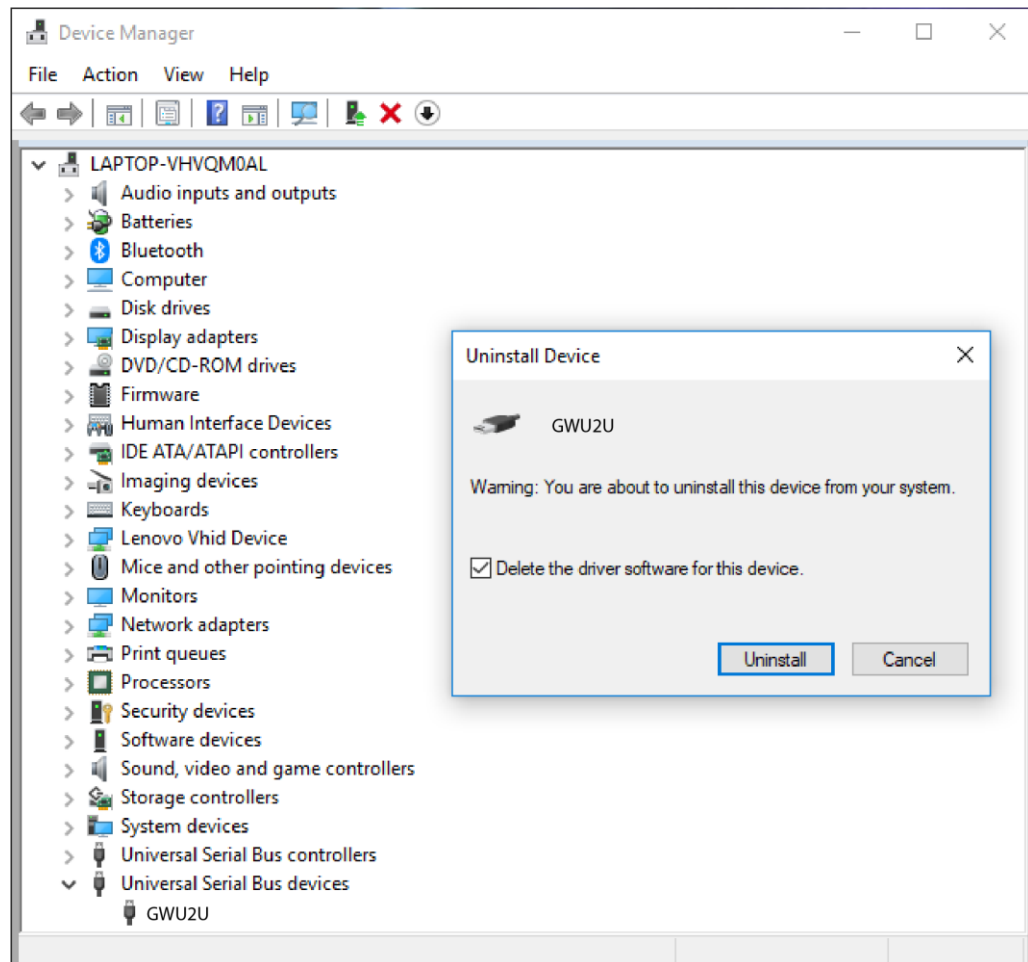
To uninstall the driver, connect GWU2X device to the computer, open the windows device manager, and find GWU2X device in the "Universal Serial Bus Devices" list. Right-click on the device name and select the "Uninstall Device" option in the pop-up menu.

Figure 2-1 Open Device Manager



In the pop-up dialog box, first check "Remove driver software for this device", and then click the "Uninstall" button to uninstall the driver.

Figure 2-2 Uninstall Device



3 libusb Programming Guide

libusb is an open source USB function library.

The official website is: <https://libusb.info>

The source code is hosted on github at: <https://github.com/libusb/libusb>

You can download the pre-compiled version, the official GCC version and VS version, including dynamic and static libraries, through the official website. You can also download the source code through github and compile it on your own.

For libusb function are as, see the official reference at:

<http://libusb.sourceforge.net/api-1.0>

3.1 Parameter Configuration

Use a `usb2uart_config_setting` structure to configure UART parameters.

The structure is defined as below.

```
typedef struct _usb2uart_config_setting {  
    unsigned char BaudRate;  
    unsigned char StopBits;  
    unsigned char Parity;  
    unsigned char DataBits;  
};
```

BaudRate: Serial communication baud rate

StopBits: Serial communication stop bit

The meanings of the different values of StopBits are as follows:

```
#define STOPBITS_1      0

#define STOPBITS_1_5    1

#define STOPBITS_2      2
```

Parity: Serial communication parity

The meanings of the different values of Parity are as follows:

```
#define GW_PARITY_NONE    0    //No parity
#define GW_PARITY_NONE    1    //Odd parity
#define GW_PARITY_NONE    2    //Even parity
#define GW_PARITY_MARK    3    //The parity bit is always 1
#define GW_PARITY_SPACE   4    //The parity bit is always 0
```

DataBits: Serial data bit

3.2 SW API Introduction

```
int usb2uart_set_config(libusb_device_handle * devh,
usb2uart_config_setting * pusb2uart_config, unsigned char
interface, unsigned int uiTimeout);
```

Use UART parameter configuration structure to configure the parameters of the USB2UART device.

Parameter:

devh: USB device operation handle

pusb2uart_config: Pointer to the structure that holds the UART parameter configuration

Interface: GWU2U device interface

uiTimeout: Timeout paramete

Return Value:

If executed correctly, 0 is returned; if an error occurs, an error code less than zero is returned.

```
int print_usb2uart_config(usb2uart_config_setting *
pusb2uart_config);
```

Print the contents of the UART parameter configuration structure.

Parameter:

pusb2uart_config: Pointer to the structure that holds the UART

parameter configuration

Return Value:

If executed correctly, 0 is returned; if an error occurs, an error code less than zero is returned.

```
int usb2uart_send_data(libusb_device_handle * devh, unsigned
char endpoint_out, int DataCnt, unsigned char * TransData, unsigned
int uiTimeout);
```

Synchronous Data Transmit

Parameter:

devh: USB device operation handle

Endpoint_out: GWU2U device output endpoint

DataCnt: The count of the data bytes to be transmitted

TransData: Pointer to the storage address of the data to be sent

uiTimeout: Timeout paramete

Return Value:

If executed correctly, 0 is returned; if an error occurs, an error code less than zero is returned.

```
int usb2uart_receive_data(libusb_device_handle * devh,
unsigned char endpoint_in, int * DataCnt, unsigned char * TransData,
unsigned int uiTimeout);
```

Synchronous Data Transmit

Parameter:

devh: USB device operation handle

endpoint_in: GWU2U device input endpoint

DataCnt: Pointer to the storage address of the data to be received

TransData: Pointer to the storage address of the data to be receive

uiTimeout: Timeout paramete

Return Value:

If executed correctly, 0 is returned; if an error occurs, an error code less than zero is returned.

3.3 libusb Function Library Development Flow

3.3.1 libusb Initialization and Exit

When programming with libusb, you need to call the function `libusb_init()` to initialize it first, and at the end of use, you should call the function `libusb_exit()` to exit it from the system.

Function declarations are as follows:

```
int libusb_init (libusb_context ** context)

void libusb_exit (libusb_context * ctx)
```

The parameter `libusb_context` is a libusb context structure that saves some configuration parameters for libusb. If `libusb_context` is not specified, a default context struct will be created, or if one already exists, it will be used directly and not reinitialized.

Programming examples are as follows:

```
int rc = libusb_init(NULL);

if (rc != LIBUSB_SUCCESS)

{

    return rc;

}

libusb_exit(NULL);
```

3.3.2 Open the Specified USB Device

You can use the `libusb_open_device_with_vid_pid()` function to open the specified device based on Vendor ID and Product ID. You can also use the `libusb_get_device_list()` function to get all the USB devices, select the desired device from them, and use the `libusb_open()` function to get the handle of the device for subsequent operations. The function declaration is as follows:

(1) Open the device with Vendor ID and Product ID:

```
libusb_device_handle*
libusb_open_device_with_vid_pid(

    libusb_context * ctx,

    uint16_t vendor_id,
```

```

        uint16_t product_id
    )

```

The parameter `ctx` is the address of the context structure generated when initializing libusb. If the default context is used, use `NULL`. `vendor_id` and `product_id` are the Vendor ID and Product ID of the USB device, respectively. The Vendor ID of the Gowin USB device is 0x33aa, and the Product ID of the GWU2U device is 0x0020.

The return value is the pointer to the operation handle of the first matching device found by libusb on this computer, otherwise it returns the null pointer `NULL`.

Examples of use are as follows:

```

devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0020);
if(NULL == devh)
{
    printf("Open USB device failed\n");
    goto out;
}
else
{
    printf("Open USB device\n");
}

```

(2) Select the specified device after getting all USB devices.

```

ssize_t libusb_get_device_list (
    libusb_context * ctx,
    libusb_device *** list
)

```

The parameter `ctx` is the address of the context struct generated when initializing libusb. If the default context is used, use `NULL`. “list” is the pointer to storage device list. At the end of use, the memory should be freed using the `libusb_free_device_list()` function.

If the function is executed correctly, the return value is the number of devices and the list saves the list of found devices. Otherwise, a `libusb_error` value less than zero is returned.

```
int libusb_open (
    libusb_device *dev,
    libusb_device_handle **dev_handle
)
```

The parameter `dev` is the device in the device list, and `dev_handle` is the address that saves the pointer of the returned device handle.

If the device is opened successfully, the return value is `LIBUSB_SUCCESS`; otherwise an `libusb_error` value less than zero is returned.

Examples of use are as follows:

```
cnt = libusb_get_device_list(NULL, &devs);
if(cnt < 0)
{
    // get device list failed
    return -1;
}
for(int i = 0; i < cnt; i++)
{
    libusb_open(dev[i], dev_handle);
    if(/*the wanted device is opened*/)
    {
        break;
    }
    else
    {
        //the current device is not wanted, close it and check the
        next one.
        libusb_close(dev_handle);
    }
}
```


3.3.3 Interface Declaration

USB devices usually contain one or more interfaces. libusb needs to declare the interface first when using the interface, and when the declaration is successful, it means that the interface is successfully opened and the endpoint contained in the interface can be received/transmitted.

```
int libusb_claim_interface(  
  
libusb_device_handle * dev_handle,  
  
int interface_number  
  
)
```

The parameter `dev_handle` is the device handle; `interface_number` is the number of interface. In GWU2U device, the interface number is 0. If the interface is declared successfully, the return value is `LIBUSB_SUCCESS`; otherwise an `libusb_error` value less than zero is returned.

Programming Examples:

```
rc = libusb_claim_interface(devh, 0);  
if (rc != LIBUSB_SUCCESS)  
{  
    printf("Error claiming interface: %s\n",  
libusb_error_name(rc));  
    goto out;  
}  
else  
{  
    printf("Claiming interface\n");  
}
```

3.3.4 Serial Communication Parameters Configuration

To set the baud rate, parity bit, stop bit, etc. for GWU2U serial devices, please use the SW API `usb2uart_set_config()` function.

Programming Examples:

```
usb2uart_config.BaudRate = 115200;  
usb2uart_config.Parity = GW_PARITY_NONE;
```

```
usb2uart_config.StopBits = STOPBITS_1;

usb2uart_config.DataBits = 8;

rc = usb2uart_set_config(devh, &usb2uart_config,
UART_INTERFACE, 0);

if(rc != 0)
{
    printf("Error : %s", libusb_strerror(rc));
    return -1;
}
```

3.3.5 Synchronous Data Transmit

For GWU2U synchronous data transmit, please use SW API `usb2uart_send_data()` function.

Programming Examples:

```
rc = usb2uart_send_data(devh, ENDPOINT_OUT, sizeof(tx_data),
tx_data, 1000);

if(rc != 0)
{
    printf("Error: %s", libusb_strerror(rc));
}
else
{
    printf("GWU2U device sends %d bytes data, return code: %d\n",
(int)(sizeof(tx_data)), rc);
}
```

3.3.6 Synchronous Data Receive

For GWU2U synchronous data receive, please use SW API `usb2uart_receive_data()` function.

Programming Examples:

```
rc = usb2uart_receive_data(devh, ENDPOINT_IN, &size, rx_data,
1000);
```

```
if(rc != 0)
{
    printf("Error: %s", libusb_strerror(rc));
}
else
{
    printf("GWU2U device receives %d bytes data, return
code: %d\n", size, rc);
    size = 0;
}
```

3.3.7 Asynchronous Data Transfer

libusb enables asynchronous data transfer.

The implementation is as follows:

- (1) Use `libusb_alloc_transfer()` function to allocate a transfer;
- (2) Use `libusb_fill_bulk_transfer()` function to fill transfer content and register the callback function;
- (3) Use `libusb_submit_transfer()` function to submit transfer request;
- (4) At the end of the transfer, call the callback function automatically to handle the result of the transfer;
- (5) At the end of the transfer, use `libusb_free_transfer()` function to free the memory.

Take asynchronous data receive as an example:

```
static uint8_t buf[PACK_SIZE];

static struct libusb_transfer *xfr; //To define transfer struct

xfr = libusb_alloc_transfer(0); //Allocate a transfer, bulk
transfer, then the parameter is 0

if (!xfr)
{
    return -ENOMEM;
}
```

```
libusb_fill_bulk_transfer(  
    xfr, //Transfer the defined struct  
    devh, //Device operation handle  
    ep, //Transfer endpoint  
    buf, //buffer to receive data  
    sizeof(buf), //The length of receive buffer  
    cb_xfr_in, //Registered callback function  
    NULL, //The pointer to the data transferred to the callback  
function, of type void  
    0// timeout parameter (in milliseconds)  
);  
  
libusb_submit_transfer(xfr);  
  
while(1)  
{  
    rc = libusb_handle_events(NULL); //Handle hang-on events  
}
```

4 Programming Example

A simple GWU2U serial port programming example is shown below.

First send a piece of data using the synchronous transmit mode, then enter the asynchronous receive mode and output the received data to the console in character form.

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include "libusb.h"

#include "gw_usb2uart.h"

static struct libusb_device_handle *devh = NULL;

unsigned char tx_data[] = "This is a usb2uart test string sent
through usb\r\n";

unsigned int rd_cnt = 0;

usb2uart_config_setting usb2uart_config;

//For callback function of asynchronous receive, when data is
received, output the data to the console as a string

void LIBUSB_CALL cb_xfr_in(struct libusb_transfer *xfr)
{
    int i;

    rd_cnt++;

    if (xfr->status != LIBUSB_TRANSFER_COMPLETED)
    {
        fprintf(stderr, "transfer status %d\n", xfr->status);
```

```
        libusb_free_transfer(xfr);
        exit(3);
    }
    for (i = 0; i < xfr->actual_length; i++)
    {
        printf("%c", xfr->buffer[i]);
    }

    //At the end of receiving data, resubmit the transfer request
    and wait for the next receive event. If the submission fails, then
    free the memory.
    if(libusb_submit_transfer(xfr) < 0)
    {
        libusb_free_transfer(xfr);
    }
}

static int benchmark_in(uint8_t ep)
{
    static uint8_t buf[PACK_SIZE];
    static struct libusb_transfer *xfr;
    printf("(SUBMIT FUNC) Starting a new asynchronous bulk
transfer\n");
    printf("Receive data :\n");
    xfr = libusb_alloc_transfer(0);
    if (!xfr)
    {
        return -ENOMEM;
    }
    libusb_fill_bulk_transfer(
        xfr,
        devh,
        ep,
        buf,
```

```
        sizeof(buf),
        cb_xfr_in,
        NULL,
        0
    );
    return libusb_submit_transfer(xfr);
}

int main(int argc, char *argv[])
{
    int rc = 0;
    //Initializes libusb
    rc = libusb_init(NULL);
    if (rc != LIBUSB_SUCCESS)
    {
        return rc;
    }
    //Open USB device
    devh = libusb_open_device_with_vid_pid(NULL, VENDOR_ID,
PRODUCT_ID);
    if(NULL == devh)
    {
        printf("Open USB device failed\n");
        goto out;
    }
    else
    {
        printf("Open USB device\n");
    }
    rc = libusb_claim_interface(devh, UART_INTERFACE);
    if (rc != LIBUSB_SUCCESS)
    {
```

```
        printf("Error claiming interface: %s\n",
libusb_error_name(rc));
        goto out;
    }
    else
    {
        printf("Claiming interface\n");
    }
    printf("DEVICE READY...\n\n");
    //Configuration settings
    usb2uart_config.BaudRate = 115200;
    usb2uart_config.Parity = 0;
    usb2uart_config.StopBits = 0;
    usb2uart_config.DataBits = 8;
    //Set configurations
    rc = usb2uart_send_data(devh, &usb2uart_config,
UART_INTERFACE, 0);
    if(rc != 0)
    {
        printf("Error: %s", libusb_strerror(rc));
        return -1;
    }
    //Send data synchronously
    rc = usb2uart_send_data(devh, ENDPOINT_OUT, sizeof(tx_data),
tx_data, 1000);
    if(rc != 0)
    {
        printf("Error: %s", libusb_strerror(rc));
    }
    else
    {
        printf("GWU2U device sends %d bytes, return code: %d\n",
```



```
(int) (sizeof(tx_data)), rc);  
  
    }  
  
    benchmark_in(ENDPOINT_IN);  
  
    while(1)  
    {  
        rc = libusb_handle_events(NULL);  
    }  
  
    libusb_release_interface(devh, UART_INTERFACE);  
  
out:  
    if(devh)  
    {  
        libusb_close(devh);  
    }  
  
    libusb_exit(NULL);  
    return 0;  
}
```

Terminology and Abbreviations

The abbreviations and terminology used in this manual are as shown in Table A-1.

Table A-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter

Platform Supported

The platforms supported by GWU2U SW API are as shown in Table A-2.

Table A-2 Platform Supported

Windows	Windows 7/10(32bit/64bit)
Linux	Ubuntu 20.04 LTS

Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

