GWU2U Driver (libusb+WinUSB based)

用户指南

UG1006-1.0, 2021-06-29

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2021/06/29	1.0	初始版本。

目录

图目录

表目录

1 GWU2U 的 libusb+WinUSB 驱动

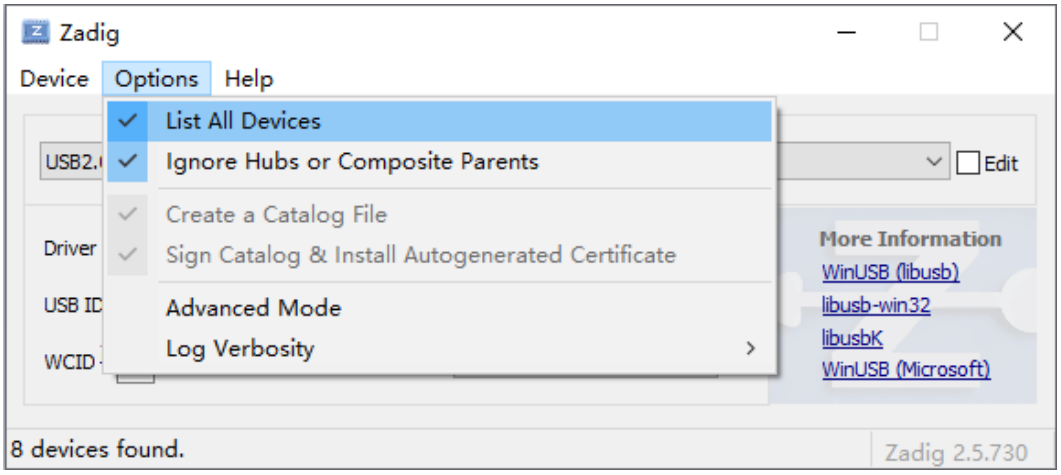
GWU2U 可使用开源 USB 函数库 libusb 进行编程操作。使用该函数库编程时，需安装 WINDOWS USB 驱动程序 WinUSB.sys。可实现数据的收发，以及波特率、校验位、停止位、字节位宽等参数的设置。

安装驱动，可使用 Gowin 提供的驱动安装工具，亦可使用开源驱动安装工具 Zadig（https://zadig.akeo.ie/）。安装驱动需要使用管理员权限。

1.1 使用 Zadig 安装驱动程序

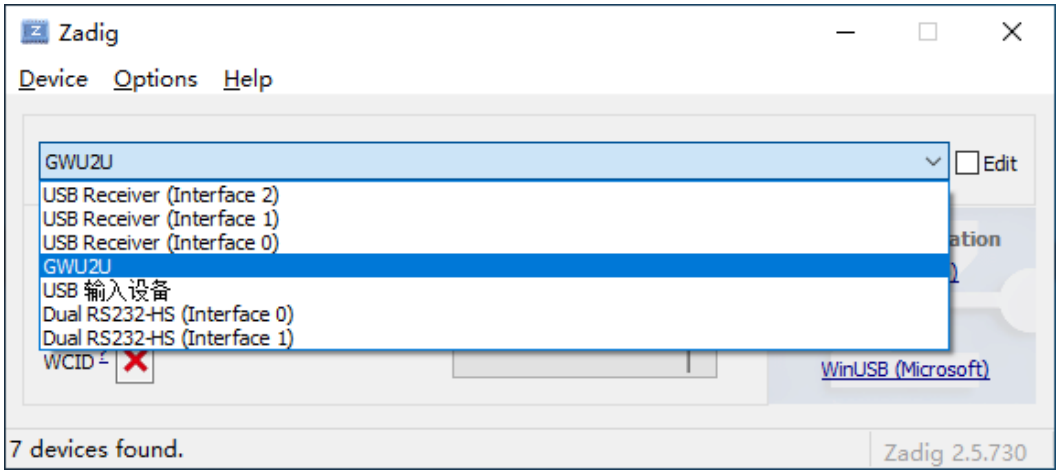
连接 GWU2U 设备到电脑 USB 接口，双击打开 Zadig（需要管理员权限），点击 Options，勾选“List All Device”选项，此时会列举出所有连接到电脑的 USB 设备。

图 1-1 列出所有 USB 设备



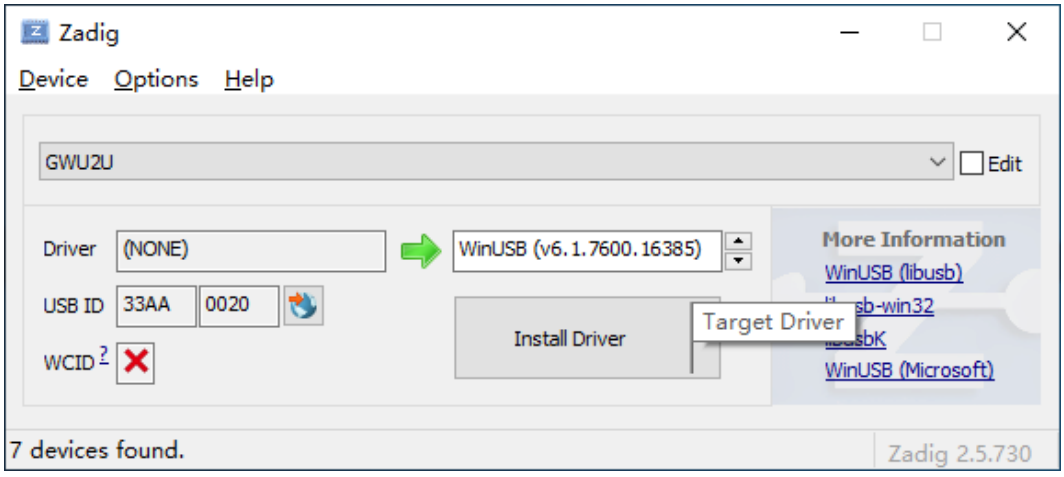
选择需要安装驱动的设备 Gowin U2U。

图 1-2 选择需要安装驱动的设备



选择要安装的驱动程序，使用 libusb+WinUSB 的形式，请选择 WinUSB。

图 1-3 选择要安装的驱动程序



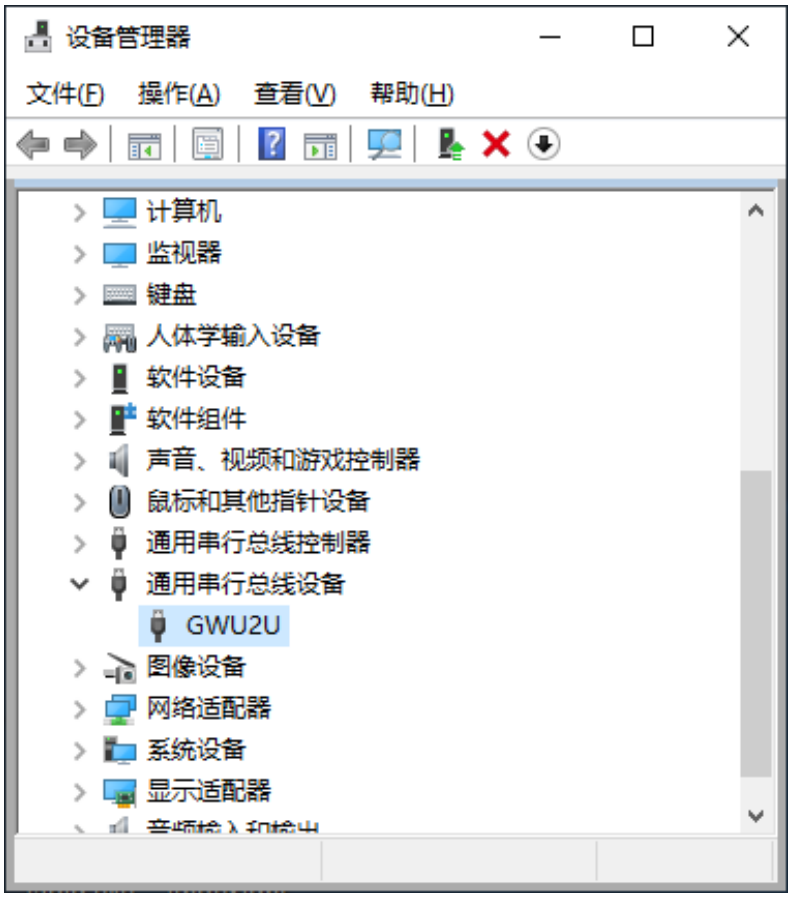
点击“Install Driver”按钮安装驱动。稍等片刻即可完成驱动安装。

需要注意的是，若当前未安装驱动，该按钮显示为“Install Driver”，若当前安装了其他驱动，则显示为“Replace Driver”。

2 驱动程序的卸载

卸载驱动程序时，连接 GWU2U 设备到电脑，打开 windows 设备管理器，在“通用串行总线设备”列表中，找到 Gowin USB Serial 设备。右单击该设备名称，在弹出菜单中选择“卸载设备”选项。

图 2-1 打开设备管理器



在弹出的对话框中，先勾选“删除此设备的驱动程序软件”，再点击卸载按钮，即可卸载驱动程序。

图 2-2 卸载设备

3 libusb+WinUSB 编程指南

libusb 是一个开源的 USB 函数库。

其官方网址是 https://libusb.info

其源代码已托管至 github：https://github.com/libusb/libusb

您可以通过其官方网址下载预编译版本，官方 GCC 版本和 VS 版本，包括动态库和静态库。亦可通过 github 下载源码并自行编译。

libusb 函数声明，请见官方参考：

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

3.1 libusb 函数库的初始化和退出

使用 libusb 编程时，需先调用函数 libusb_init()对其进行初始化，使用结束后，应当调用函数 libusb_exit()令其从系统中退出。

函数声明如下：

```
int libusb_init (libusb_context ** context)

void libusb_exit (libusb_context * ctx)
```

其中，参数 libusb_context 是一个 libusb 上下文结构体，用于保存 libusb 的一些配置参数。如果不指定 libusb_context，则会创建一个默认的上下文结构，若已存在一个上下文结构，则直接使用该结构且不重新初始化。

编程举例如下：

```
int rc = libusb_init(NULL);

if (rc < 0)

return rc;

libusb_exit(NULL);
```

3.2 打开指定的 USB 设备

可使用 libusb_open_device_with_vid_pid()函数，根据 VID/PID 打开指定设备。也可使用 libusb_get_device_list()函数，获取所有的 USB 设备，从中选出所需的设备，使用 libusub_open()函数获取该设备的句柄，用于进行后续的操作，函数声明如下：

(1) 使用 VID/PID 打开设备：

```
libusb_device_handle* libusb_open_device_with_vid_pid(
                        libusb_context * ctx,
                        uint16_t vendor_id,
                        uint16_t product_id
                    )
```

其中，参数 ctx 为初始化 libusb 时产生的上下文结构的地址，若使用默认上下文，则使用 NULL；vendor_id 和 product_id 分别为 USB 设备的 VID 和 PID，Gowin USB 设备的 VID 为 0x33aa，GWU2U 设备的 PID 为 0x0020。

返回值为 libusb 在此计算机上查找到的第一个符合的设备的操作句柄指针，否则返回空指针 NULL。

使用举例如下：

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

(2) 获取所有 USB 设备后选取指定设备：

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

其中，参数 ctx 为初始化 libusb 时产生的上下文结构的地址，若使用默认上下文，则使用 NULL；list 为存储设备列表的指针。在使用结束后，应当使用 libusb_free_device_list（）函数释放该段内存。

若函数执行正确，则返回值为设备数量，并且 list 中保存查找到的设备列表。否则，返回一个小于零的 libusb_error 值。

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

其中，参数 dev 为设备列表中的设备，dev_handle 为保存返回设备句柄的指针的地址。

若设备打开成功，则返回值为 0；否则，返回一个小于零的 libusb_error 值。

使用举例如下：

```
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 接口声明

USB 设备通常包含一个或多个接口（interface），libusb 在使用接口时，需先声明接口（claim interface），声明成功后，则表示该接口被成功打开，可以对该接口包含的端点（endpoint）进行收发操作。

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

其中，参数 dev_handle 是设备句柄，interface_number 是接口编号。在 GWU2U 设备中，接口编号为 0。若声明接口成功，则返回 0，否则返还一个小于零的 libusb_error 值。

编程举例：

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

3.4 串口参数设置

GWU2U 串口设备设置波特率、校验位、停止位等操作，需要通过控制端点（endpoint0）传输控制指令。Libusb 提供函数 libusb_control_transfer()函数用于传输控制指令：

```
int libusb_control_transfer (
libusb_device_handle * dev_handle,
uint8_t bmRequestType,
uint8_t bRequest,
uint16_t wValue,
uint16_t wIndex,
unsigned char * data,
uint16_t wLength,
unsigned int timeout
```

```
)
```

其中，参数 dev_handle 为设备的操作句柄；bmRequestType、bRequest、wValue、wIndex 是控制命令的参数，在设置 GWU2U 串口参数时，bmRequestType 的值为 0b00100001，bRequest 的值为 0x20，wValue 和 wIndex 的值应为 0。

Data 是一个指向保存控制指令数据部分的指针，而 wLength 则是控制指令数据部分的长度。

在设置 GWU2U 串口参数时，控制指令的数据部分是一个保存串口参数的结构体。该结构体定义如下：

```
typedef struct _vcp_config_setting {
   unsigned int  BaudRate;   //波特率
   unsigned char StopBits;   //停止位
   unsigned char Parity;     //校验位
   unsigned char DataBits;   //字节位数
} vcp_config_setting;
```

其中，StopBits 不同取值的含义如下：

```
#define STOPBITS_1    0
#define STOPBITS_1_5  1
#define STOPBITS_2    2
```

Parity 不同取值的含义如下：

```
#define VCP_PARITY_NONE   0
#define VCP_PARITY_ODD    1
#define VCP_PARITY_EVEN   2
#define VCP_PARITY_MARK   3
#define VCP_PARITY_SPACE  4
```

编程举例：

```
rc = libusb_control_transfer(
devh, 0b00100001, 0x20, 0, 0,
(unsigned char *)(&vcp_config), sizeof(vcp_config), 0
);
```

3.5 同步数据发送

使用批量传输(bulk transfer)，可实现 GWU2U 的数据发送和接收。libusb 提供一个 libusb_bulk_transfer()函数以实现对指定端点数据的批量传输，传输方向由端口编号确定。在 GWU2U 设备中，端口 0x02 为输出端口，用于 GWU2U 发送数据。

```
int libusb_bulk_transfer (
libusb_device_handle * dev_handle,
      unsigned char endpoint,
      unsigned char *data,
      int length,
      int *transferred,
      unsigned int timeout
   )
```

其中，参数 dev_handle 为设备操作句柄，endpoint 为端口编号，执行数据发送操作时，端口编号应为 0x02，data 为保存待发送数据的地址，length 为待发送数据的长度，transferred 为保存返回的实际发送数据长度的变量地址，timeout 为超时控制变量（单位为毫秒），表示放弃操作前等待的时长，若设置为 0，则为无限等待。

若数据传输成功，则返回值为 0，否则返回一个小于零的 libusb_error 值。

编程举例：

```
rc = libusb_bulk_transfer(devh, 0x02, tx_data, sizeof(tx_data),
&size, 1000);
if(rc < 0) {
   printf("Error: %s", libusb_strerror(rc));
}else {
   printf("%d bytes written, ret: %d\n", size, rc);
   size = 0;
}
```

3.6 同步数据接收

与同步数据发送相类似，亦是使用 libusb_bulk_transfer()函数进行数据接收。接收的端口号为 0x82。

编程举例：

```
rc = libusb_bulk_transfer(devh, 0x82, tx_data, sizeof(tx_data), &size, 1000);
if(rc < 0) {
    printf("Error: %s", libusb_strerror(rc));
}else {
    printf("%d bytes received, ret: %d\n", size, rc);
    size = 0;
}
```

3.7 异步数据传输

libusb 可实现异步数据的传输。实现方法如下：

(1) 使用 libusb_alloc_transfer()函数分配一个传输；

(2) 使用 libusb_fill_bulk_transfer()函数填充传输内容并注册回调函数；

(3) 使用 libusb_submit_transfer()函数提交传输请求；

(4) 当传输完成后，自动调用回调函数，对传输的结果进行处理。

(5) 传输结束后，需使用 libusb_free_transfer()函数释放内存。

以异步数据接收为例：

```
static uint8_t buf[PACK_SIZE];
static struct libusb_transfer *xfr;  //用于定义传输的结构
xfr = libusb_alloc_transfer(0);  //分配一个传输，bulk 传输，则参数为 0
if (!xfr)
    return -1;
libusb_fill_bulk_transfer(
    xfr,  //传输定义结构
    devh, //设备操作句柄
    ep,  //传输端点
```

```
    buf, //接收数据的 BUFFER
    sizeof(buf), //接收 BUFFER 的长度
    cb_xfr_in, //注册的回调函数
    NULL, //传递给回调函数的数据的指针，类型为 void
    0 //超时定义（单位为毫秒）
);
libusb_submit_transfer(xfr);
while(1) {
rc = libusb_handle_events(NULL); //处理挂起的事件
}
```

4编程举例

下面是一个简单的 GWU2U 串口编程示例。先使用同步发送模式发送一条数据，然后进入异步接收模式，并将接收到的数据以字符形式输出到控制台。

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <windows.h>
#include <iostream>
#include "libusb.h"

#define TEST_SIZE 50
#define STOPBITS_1    0
#define STOPBITS_1_5  1
#define STOPBITS_2    2
#define VCP_PARITY_NONE   0
#define VCP_PARITY_ODD    1
#define VCP_PARITY_EVEN   2
#define VCP_PARITY_MARK   3
#define VCP_PARITY_SPACE  4

typedef struct _vcp_config_setting {
   unsigned int  BaudRate;
   unsigned char StopBits;
```

```
    unsigned char Parity;
    unsigned char DataBits;
} vcp_config_setting;

static struct libusb_device_handle *devh = NULL;

#define PACK_SIZE   64

uint8_t endpoint_in  = 0x82;
uint8_t endpoint_out = 0x02;  // default IN and OUT endpoints

unsigned char tx_data_tmp[TEST_SIZE];
unsigned char rx_data_tmp[TEST_SIZE * 2];
int size = 0;

uint8_t endpoint_in_array[1]  = {0x82};
uint8_t endpoint_out_array[1] = {0x02};

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

static unsigned long num_bytes = 0, num_xfer = 0;
unsigned int wr_cnt = 0;
unsigned int rd_cnt = 0;

vcp_config_setting vcp_config;
vcp_config_setting vcp_config_back;

//异步接收的回调函数，当接收到数据后，以字符串的形式输出到控制台
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]);
        }
         // 本次接收数据完成后，重新提交传输请求，并等待下次接收事件，若提交失败，则释放该段内存
        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 async bulk transfer, READ\n\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;
        int i = 0;

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

        rc = libusb_claim_interface(devh, 0);
        if (rc < 0) {
            printf("Error claiming interface: %s\n",
libusb_error_name(rc));
```

```
        goto out;
    }
    printf("DEVICE READY...\n\n");

    vcp_config.BaudRate  = 115200;
    vcp_config.Parity    = 0;
    vcp_config.StopBits  = 0;
    vcp_config.DataBits  = 8;
    rc = libusb_control_transfer (
devh, 0b00100001, 0x20, 0, 0,
(unsigned char *)(&vcp_config), sizeof(vcp_config), 0
);
    if(rc < 0) {
        printf("Error: %s", libusb_strerror(rc));
        return -1;
    }

    rc = libusb_bulk_transfer(
devh, endpoint_out, tx_data, sizeof(tx_data),
&size, 1000
);
    if(rc < 0) {
        printf("Error: %s", libusb_strerror(rc));
    }
    else {
        printf("%d bytes written, ret: %d\n", size, rc);
        size = 0;
    }

    benchmark_in(endpoint_in);
    while(1) {
```

```
            rc = libusb_handle_events(NULL);
        }

        libusb_release_interface(devh, 0);
    out:
        if(devh)
            libusb_close(devh);
        libusb_exit(NULL);
        return 0;
    }
```

术语、缩略语

表 A-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 A-1 术语、缩略语

术语、缩略语	全称	含义
USB	Universal Serial Bus	通用串行总线
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发器

技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391


GOWIN高云
智慧逻辑 定制未来