



# GWU2X 编程指南\_U2X\_IIC

UG1002-1.0, 2021-06-29

版权所有 © 2021 广东高云半导体科技股份有限公司

 GOWIN高云, Gowin, 高云均为广东高云半导体科技股份有限公司注册商标, 本手册中提到的其他任何商标, 其所有权利属其所有者所有。未经本公司书面许可, 任何单位和个人都不得擅自摘抄、复制、翻译本档内容的部分或全部, 并不得以任何形式传播。

### 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

目录 .....	i
表目录.....	ii
<b>1 功能简介 .....</b>	<b>1</b>
<b>2 驱动的安装和卸载.....</b>	<b>2</b>
2.1 使用 Zadig 安装驱动程序 .....	2
2.2 卸载驱动程序 .....	4
<b>3 libusb_WinUSB 的编程指南.....</b>	<b>5</b>
3.1 libusb 函数库的初始化和退出 .....	5
3.2 打开指定的 USB 设备 .....	5
3.3 接口声明.....	9
<b>4 U2X_IIC 的 API 函数 .....</b>	<b>8</b>
4.1 U2X_IIC 初始化 .....	8
4.2 数据发送.....	8
4.3 数据接收.....	9
4.4 编程示例.....	11
<b>5 错误代码 .....</b>	<b>13</b>
术语、缩略语.....	14
技术支持与反馈 .....	15

## 图目录

图 2-1 选择“List All Device”选项 .....	2
图 2-2 选择需要安装驱动的设备 .....	3
图 2-3 选择要安装的驱动程序 .....	3
图 2-4 打开设备管理器 .....	4
图 2-5 卸载设备 .....	4

# 表目录

表 5-1 错误代码列表 .....	13
表 A-1 术语、缩略语 .....	14

# 1 功能简介

GWU2X 是一个 USB 转多种协议的转换器，可实现 USB2IIC 功能转换，支持最高 SCL 时钟频率 500KHz（当前的测试结果）。

支持 IIC 的主机模式的数据收发功能，支持 7-bit 地址模式和 10-bit 地址模式。

# 2 驱动的安装和卸载

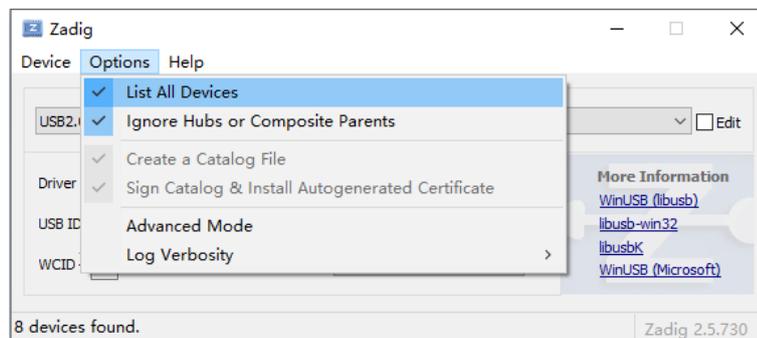
GWU2X\_IIC 可使用开源 USB 函数库 libusb 进行编程操作。使用该函数库编程时，需安装 WINDOWS USB 驱动程序 WinUSB.sys。

安装驱动，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。

## 2.1 使用 Zadig 安装驱动程序

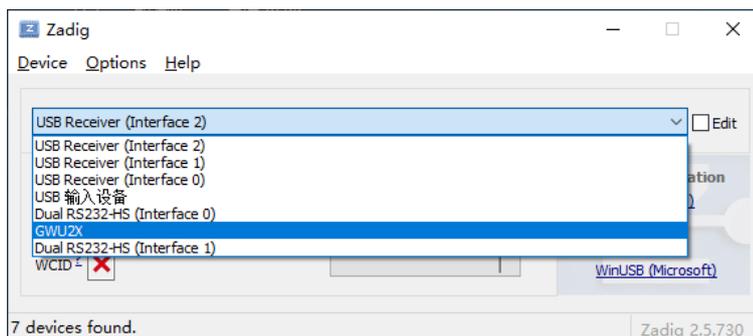
连接 GWU2X 设备到电脑 USB 接口，双击打开 Zadig(需要管理员权限)，单击 Options，勾选“List All Device”选项，此时会列举出所有连接到电脑的 USB 设备。

图 2-1 选择“List All Device”选项



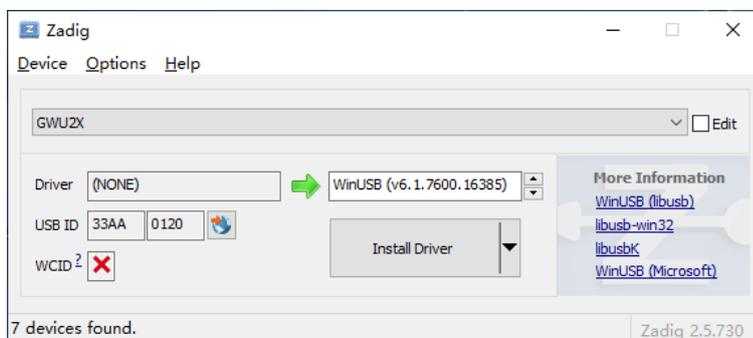
选择需要安装驱动的设备 GWU2X。

图 2-2 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 2-3 选择要安装的驱动程序



点击“Install Driver”<sup>[1]</sup>按钮安装驱动。稍等片刻即可完成驱动安装。

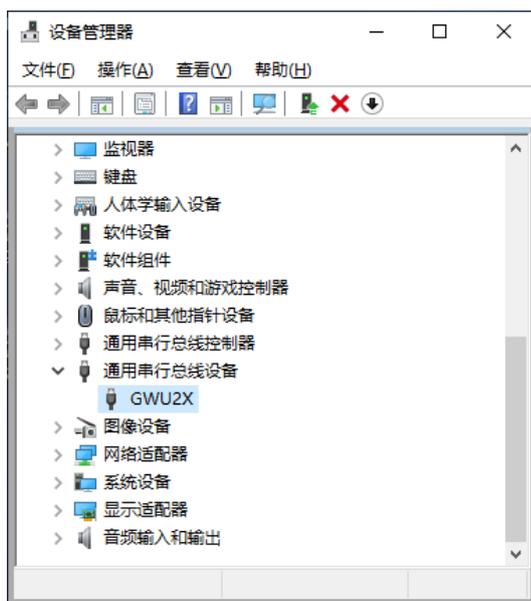
注！

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”。

## 2.2 卸载驱动程序

卸载驱动程序时，连接 GWU2X 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 GWU2X 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 2-4 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 2-5 卸载设备



# 3 libusb\_WinUSB 的编程指南

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>

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 函数说明，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 `libusb_open()` 函数获取该设备的句柄，用于进行后续的操作，函数说明如下：

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，USB2IIC 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;  
}
```

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);

    }

}
```

# 4 U2X\_IIC 的 API 函数

## 4.1 U2X\_IIC 初始化

该函数用于 U2X\_IIC 的初始化，以及 SCL 时钟频率的配置，若需要修改 SCL 时钟频率，需重新调用该函数进行配置。

```
int u2x_iic_init(  
  
libusb_device_handle *devh,  
  
unsigned int uiFreqKiloHz,  
  
unsigned int uiTimeout);
```

函数参数:

- devh: libusb 的设备操作句柄;
- uiFreqKiloHz: SCL 时钟频率，以千赫兹为单位;
- uiTimeout: 超时参数，单位为毫秒;

返回值:

若运行成功，则返回值为 0，否则返回一个小于零的错误代码。

## 4.2 数据发送

通过 IIC 向指定的 IIC Slave 设备发送数据，单次发送数据长度最大为 256 字节。

```
int u2x_iic_send_bytes(  
  
libusb_device_handle *devh,  
  
unsigned int uiAddr,  
  
unsigned int uiAddrBit,  
  
unsigned char *pucData,
```

```
unsigned int uiDataLen,  
unsigned int uiTimeout);
```

函数参数:

- **devh**: libusb 的设备操作句柄;
- **uiAddr**: IIC Slave 设备的地址;
- **uiAddrBit**: 地址位数, 若使用 7BIT 地址, 则该参数设置为 7; 若使用 10BIT 地址, 则该参数设置为 10; 其他值为不可用的值;
- **pucData**: 指针, 指向保存要发送数据的内存地址;
- **uiDataLen**: 要发送数据的字节数;
- **uiTimeout**: 超时参数, 单位为毫秒;

返回值:

若运行成功, 则返回值为 0, 否则返回一个小于零的错误代码。

## 4.3 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**) 进行首发操作。

```
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 GWU2X 设备中, 接口编号为 0。若声明接口成功, 则返回 0, 否则返回一个小于零的 **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;  
}
```

## 4.4 数据接收

通过 IIC 向指定的 IIC Slave 设备接收数据, 单次接收数据长度最大为 256 字节。

```
int u2x_iic_read_bytes(  
libusb_device_handle *devh,  
unsigned int uiAddr,  
unsigned int uiAddrBit,  
unsigned char *pucData,  
unsigned int uiDataLen,  
unsigned int uiTimeout);
```

函数参数:

- devh: libusb 的设备操作句柄;
- uiAddr: IIC Slave 设备的地址;
- uiAddrBit: 地址位数, 若使用 7BIT 地址, 则该参数设置为 7; 若使用 10BIT 地址, 则该参数设置为 10; 其他值为不可用的值;
- pucData: 指针, 指向保存接收数据的内存地址;
- uiDataLen: 要接收数据的字节数;
- uiTimeout: 超时参数, 单位为毫秒;

返回值:

若运行成功, 则返回值为 0, 否则返回一个小于零的错误代码。

## 4.5 编程示例

```
int main(int argc, char *argv[])
{
    unsigned char txdata[TEST_BYTE];

    unsigned char rxdata[TEST_BYTE];

    int i = 0;

    int rc = 0;

    // 打开设备

    rc = libusb_init(NULL);

    if (rc < 0)

    return rc;

    devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0120);

    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;

    }

    txdata[0] = 0x55;

    for(i = 1; i < TEST_BYTE; i++) {

        txdata[i] = i;

    }

    // 初始化 U2X IIC

    u2x_iic_init(devh, 400, 1000);

    // 向地址为 0x50 的 IIC Slave 设备发送 4 字节数据, 地址为 7bit 模式

    u2x_iic_send_bytes(devh, 0x50, 7, txdata, 4, 1000);

    // 向地址为 0x50 的 IIC Slave 设备读取 4 字节数据, 地址为 7bit 模式
```

```
    u2x_iic_read_bytes(devh, 0x50, 7, rxdata, 4, 1000);  
  
    // 关闭设备并退出  
  
    libusb_release_interface(devh, 0);  
  
out:  
  
    if(devh)  
  
        libusb_close(devh);  
  
    libusb_exit(NULL);  
  
    return 0;  
  
    }
```

# 5 错误代码

API 函数运行正常时，返回值为 0，否则返回一个负数。

非 0 返回值代表的错误含义如下表所示。

**表 5-1 错误代码列表**

Value	Enumerator	
0	SUCCESS	运行正确无错误
-1	USB_ERROR_IO	USB 输入/输出错误
-2	USB_ERROR_INVALID_PARAM	USB 参数错误
-3	USB_ERROR_ACCESS	权限不足，不能访问设备
-4	USB_ERROR_NO_DEVICE	没有找到 USB 设备（设备已断开）
-5	USB_ERROR_NOT_FOUND	未找到实体（Entity）
-6	USB_ERROR_BUSY	USB 设备忙
-7	USB_ERROR_TIMEOUT	超时
-8	USB_ERROR_OVERFLOW	内存溢出
-9	USB_ERROR_PIPE	管道错误
-10	USB_ERROR_INTERRUPTED	系统函数被中断
-11	USB_ERROR_NO_MEM	内存不足
-12	USB_ERROR_NOT_SUPPORTED	当前平台不支持该操作
-13	U2X_IIC_ERROR_USBTRANS_ERR	USB 数据传输错误
-14	U2X_IIC_ERROR_INVALID_PARAM	该参数设定值不可用
-15	U2X_IIC_ERROR_TIMEOUT	U2X_IIC 传输超时
-16	U2X_IIC_ERROR_CMD_ERR	U2X_IIC 指令错误
-17	U2X_IIC_ERROR_NO_SPEC_SLV	U2X_IIC 未找到指定的从设备
-99	ERROR_OTHER	其他错误

## 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
IIC	Inter-Integrated Circuit	两线式串行总线

## 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](http://www.gowinsemi.com.cn)

E-mail：[support@gowinsemi.com](mailto:support@gowinsemi.com)

Tel: +86 755 8262 0391

