GWU2X 编程指南_U2X_JTAG

UG1003-1.0, 2021-06-16

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
2021/06/16	1.0	初始版本。

目录

目录 i

图目录 ii

表目录 iii

1 功能简介 1

2 驱动的安装和卸载 2

2.1 使用 Zadig 安装驱动程序 2

2.2 卸载驱动程序 4

3 编程指南 5

3.1 API 函数 5

3.1.1 获取 JTAG 状态的字符串 5

3.1.2 获取函数运行错误信息的字符串 5

3.1.3 生成 TMS 信号序列 5

3.1.4 TCK 频率设置 6

3.1.5 IO 端口设置 6

3.1.6 JTAG 状态复位 7

3.1.7 JTAG 状态跳转至 IDLE 8

3.1.8 发送 IR 数据 8

3.1.9 发送 DR 数据 9

3.1.10 回读 TDO 数据 10

3.1.11 生成发送 DR 数据的指令（不发送） 10

3.2 编程举例 11

3.3 错误代码 12

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 3-1 错误代码列表 12
表 A-1 术语、缩略语 14

1 功能简介

GWU2X 为 USB 转 JTAG Master 模式，可用于读写 JTAG 接口设备，其 TCK 时钟频率可进行配置。最高支持 15MHz TCK（当前测试结果）。可发送或读取任意 BIT 长度的 IR/DR 数据。

2 驱动的安装和卸载

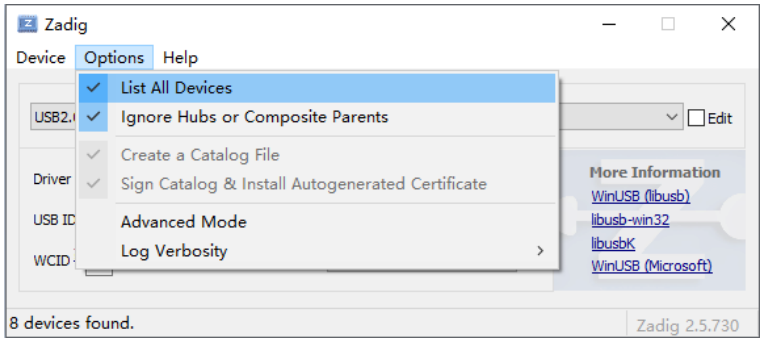
GWU2X_JTAG 可使用开源 USB 函数库 libusb 进行编程操作。使用该函数库编程时，需安装 WINDOWS USB 驱动程序 WinUSB.sys。

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

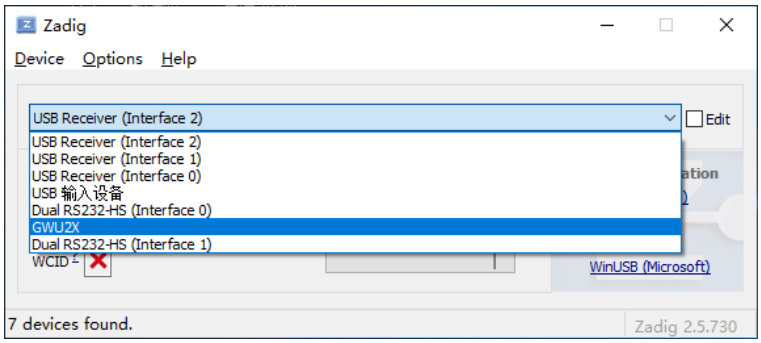
连接 GWU2X 设备到电脑 USB 接口，双击打开 Zadig（需要管理员权限），点击 Options，勾选“List All Device”选项，此时会列举出所有连接到电脑的 USB 设备。

图 2-1 选择“List All Device”选项



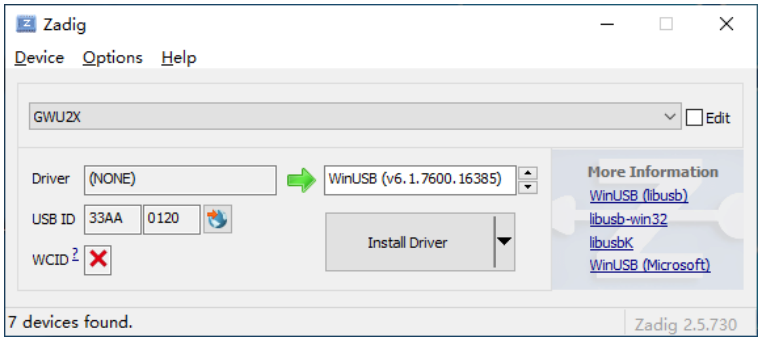
选择需要安装驱动的设备 GWU2X。

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”[1]按钮安装驱动。稍等片刻即可完成驱动安装。

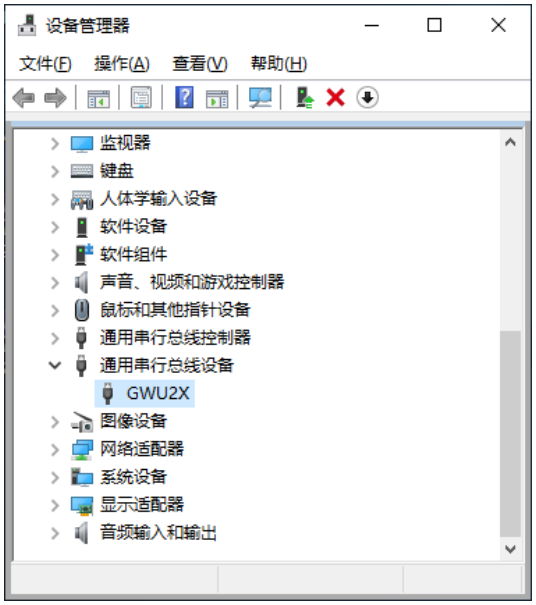
注！

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 编程指南

3.1 API 函数

3.1.1 获取 JTAG 状态的字符串

```
const char * jtag_get_state_str(int iJtagStateCode);
```

函数参数：

iJtagStateCode：表示 JTAG 状态的变量，变量定义请参考 gw_usb2jtag.h 中定义的枚举值 JTAG_STATE。

返回值：

一个 const char 型指针，指向描述 JTAG 状态的字符串的地址，可用于输出信息便于调试。

3.1.2 获取函数运行错误信息的字符串

```
const char * usb2jtag_get_err_info(int errCode);
```

函数参数：

errCode：API 函数运行返回的错误代码，若为 0，表示运行成功；若为负值，则表示函数运行出现错误。

返回值：

一个 const char 型指针，指向描述错误信息的字符串的地址，可用于输出信息便于调试。

3.1.3 生成 TMS 信号序列

```
int jtag_generate_tms_array(

int *pJtagState,
```

```
int iTargetState,

int *pTmsBitCnt,

unsigned int *pTmsBitArray);
```

函数参数：

pJtagState：一个指向暂存当前 JTAG 状态的变量的指针，若运行成功，该变量会被更新为 TMS 信号发送后的 JTAG 状态；

iTargetState：想要通过发送 TMS 信号使得 JTAG 设备跳转到的目标 JTAG 状态；

pTmsBitCnt：一个指针，生成的 TMS 信号序列的长度将保存在该指针指向的变量中；

pTmsBitArray：一个指针，生成的 TMS 信号序列的值将保存在该指针指向的变量中。

返回值：

若参数设置成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.4 TCK 频率设置

```
int usb2jtag_tckset(

libusb_device_handle *devh,

unsigned int uiFreqKiloHz,

unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

uiFreqKiloHz：要设置的 TCK 频率，单位为 KHz；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若参数设置成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.5 IO 端口设置

```
int usb2jtag_ioset(

libusb_device_handle *devh,

unsigned short usIODir,

unsigned short usIOLevelSet,
```

```
unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

uiIODir：IO 端口的方向配置。需将 JTAG TCK/TMS/TDI 设置为输出模式，TDO 设置为输入模式。其参数的定义请参考 gw_usb2jtag.h 中的宏定义。

uiIOLevelSet：设置 IO 端口在空闲时的高低电平。通常需要将 TMS/TDI 设置为空闲时高电平，TCK/TDO 设置为空闲时低电平。其参数的定义请参考 gw_usb2jtag.h 中的宏定义。

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若参数设置成功，则返回值为 0，否则返回一个小于零的错误代码。

使用举例：

```
usIODir = JTAG_GPIO_TCK_OUT | JTAG_GPIO_TMS_OUT | JTAG_GPIO_TDI_OUT;

usIOLevelSet = JTAG_GPIO_TMS_HIGH | JTAG_GPIO_TDI_HIGH;

usb2jtag_ioset(devh, usIODir, usIOLevelSet, TIMEOUT);
```

3.1.6 JTAG 状态复位

执行该函数时，JTAG 发送一串 TMS 高电平信号，使得 JTAG 设备状态跳转到 TEST LOGIC RESET。

```
int usb2jtag_reset(

libusb_device_handle *devh,

int *pJtagCurrentState,

unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

pJtagCurrentState：一个指针，指向保存当前 JTAG 状态的变量，该函数执行完成后，该变量将会变为 TEST_LOGIC_RESET (0x0)；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.7 JTAG 状态跳转至 IDLE

执行该函数时，JTAG 发送 TMS 信号，使得 JTAG 设备状态跳转至 RUN TEST IDLE。

```
int usb2jtag_goto_idle(

libusb_device_handle *devh,

int *pJtagCurrentState,

unsigned short usIdleTckLen,

unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

pJtagCurrentState：一个指针，指向保存当前 JTAG 状态的变量，该函数执行完成后，该变量将会变为 RUN TEST IDLE(0x0)；

usIdleTckLen：跳转至 RUN TEST IDLE 后，继续产生一段指定长度的 TCK，若不需要，则将该参数传递为 0；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.8 发送 IR 数据

执行该函数时，先发送 TMS 信号使得 JTAG 状态跳转至 SHIFT IR，然后发送指定的 IR 数据，发送完成后再发送 TMS 信号使得 JTAG 状态跳转至 RUN TEST IDLE，若有需要，继续发送指定周期数的 TCK 信号。

```
int usb2jtag_shift_ir(

libusb_device_handle *devh,

int *pJtagCurrentState,

int iTdiBitCnt,

unsigned char *ucTdiData,

unsigned char *ucTdoReadBack,

unsigned short usIdleTckLen,

unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

pJtagCurrentState：一个指针，指向保存当前 JTAG 状态的变量，该函数执行完成后，该变量将会变为 RUN TEST IDLE(0x0)；

iTdiBitCnt：TDI 发送 IR 数据的 BIT 长度；

ucTdiData：指向要通过 TDI 发送的 IR 数据的指针；

ucTdoReadBack：指向保存 TDO 回读数据的地址的指针，若不需要保存 TDO 回读数据，则该参数传递为 NULL；

usIdleTckLen：跳转至 RUN TEST IDLE 后，继续产生一段指定长度的 TCK，若不需要，则将该参数传递为 0；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.9 发送 DR 数据

执行该函数时，先发送 TMS 信号使得 JTAG 状态跳转至 SHIFT DR，然后发送指定的 DR 数据，发送完成后再发送 TMS 信号使得 JTAG 状态跳转至 RUN TEST IDLE，若有需要，继续发送指定周期数的 TCK 信号。

```
int usb2jtag_shift_dr(

libusb_device_handle *devh,

int *pJtagCurrentState,

int iTdiBitCnt,

unsigned char *ucTdiData,

unsigned char *ucTdoReadBack,

unsigned short usIdleTckLen,

unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

pJtagCurrentState：一个指针，指向保存当前 JTAG 状态的变量，该函数执行完成后，该变量将会变为 RUN TEST IDLE(0x0)；

iTdiBitCnt：TDI 发送 DR 数据的 BIT 长度；

ucTdiData：指向要通过 TDI 发送的 DR 数据的指针；

ucTdoReadBack：指向保存 TDO 回读数据的地址的指针，若不需要保存 TDO 回读数据，则该参数传递为 NULL；

usIdleTckLen：跳转至 RUN TEST IDLE 后，继续产生一段指定长度的

TCK，若不需要，则将该参数传递为 0；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.10 回读 TDO 数据

执行此函数时，回读当前暂存的 TDO 数据。

```
int usb2jtag_readback_tdo(
libusb_device_handle *devh,
int iBitsToRead,
unsigned char *ucTdoReadBack,
unsigned int uiTimeOut);
```

函数参数：

devh：libusb 的 USB 设备句柄，用于操作 USB2JTAG 设备；

iBitsToRead：需要回读的 TDO BIT 长度；

ucTdoReadBack：一个指针，指向保存 TDO 回读数据的地址；

uiTimeout：超时参数，单位为毫秒，若设置为 0，则为无限等待。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.1.11 生成发送 DR 数据的指令（不发送）

执行此函数时，根据传入参数生成发送 DR 数据的指令，但是不发送。可生成一组若干个发送 DR 数据指令并保存在连续地址，然后使用 libusb 的 libusb_bulk_transfer 函数一次性发送，可以提高传输效率。

```
int usb2jtag_build_shift_dr_cmd(
int *pJtagCurrentState,
int iTdiBitCnt,
unsigned char *ucTdiData,
unsigned short usIdleTckLen,
unsigned char ucIsTdoRdbk,
unsigned char *ucCmdMem,
int *piCmdMemByteLeft,
```

```
int *piCmdByteCnt);
```

函数参数：

pJtagCurrentState：一个指针，指向保存当前 JTAG 状态的变量，函数执行完成后，该变量的值将变为 RUN_TEST_IDLE。

iTdiBitCnt：要通过 TDI 发送的 DR 数据的 BIT 长度；

ucTdiData：一个指针，指向要通过 TDI 发送的 DR 数据地址；

usIdleTckLen：发送完成并跳转至 IDLE 后，需发送多少周期的 TCK，若不需要发送，则该参数传递为 0；

ucIsTdoRdbk：是否需要保存 TDO 回读数据，若需要保存，则该参数传递为 1，否则该参数传递为 0；

ucCmdMem：一个指针，指向保存生成指令的地址；

piCmdMemByteLeft：一个指针，指向保存生成指令的地址中还有多少字节空间的变量；该函数执行成功后，自动更新该变量的值；

piCmdByteCnt：一个指针，指向保存函数该次执行生成的指令的 BYTE 长度的变量。

返回值：

若函数运行成功，则返回值为 0，否则返回一个小于零的错误代码。

3.2 编程举例

```
#define TIMEOUT  1000

usIODir = JTAG_GPIO_TCK_OUT | JTAG_GPIO_TMS_OUT | JTAG_GPIO_TDI_OUT;

usIOLevelSet = JTAG_GPIO_TMS_HIGH | JTAG_GPIO_TDI_HIGH;


usb2jtag_ioset(devh, usIODir, usIOLevelSet, TIMEOUT);

usb2jtag_tckset(devh, 1330, TIMEOUT);

usb2jtag_reset(devh, &iJtagCurrentState, TIMEOUT);


usb2jtag_goto_idle(devh, &iJtagCurrentState, 0, TIMEOUT);

rc = usb2jtag_shift_ir(devh, &iJtagCurrentState, 5, &ir_data, &ir_rdbk, 10,
TIMEOUT);

if(rc != 0) {

printf("usb2jtag shift ir test failed...\n");

goto out;
```

```
    }
    printf("ir tdo read back: 0x%02x\n", ir_rdbk);

    rc = usb2jtag_shift_dr(devh, &iJtagCurrentState, 58, (unsigned char
*)(&dr_data), (unsigned char *)(&dr_rdbk), 20, TIMEOUT);
    if(rc != 0) {
    printf("usb2jtag shift dr test failed...\n");
    goto out;
    }
    printf("dr tdo read back: %016llx\n", dr_rdbk);
```

3.3 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 3-1 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	U2J_ERROR_USBTRANS_ERR	USB 数据传输错误
-14	U2J_ERROR_INVALID_PARAM	U2X_JTAG 参数设定值不可用
-15	U2J_ERROR_TIMEOUT	U2X_JTAG 传输超时

Value	Enumerator	
-16	U2J_ERROR_CMD_ERR	U2X_JTAG 指令错误
-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
JTAG	Joint Test Action Group	联合测试工作组

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